Universitat de Lleida

Escola Politècnica Superior

TREBALL FINAL DE CARRERA

ENGINYERIA TÈCNICA INFORMÀTICA DE GESTIÓ

-Pla 2001-

DESENVOLUPAMENT D'UN SISTEMA D'ADQUISICIÓ I GESTIÓ DE DADES INTEGRAT EN UNA ARQUITECTURA MODULAR EN XARXA (ORIENTAT A LA RAMADERIA DE PRECISIÓ)

Autor: Javier Casero Garcia

Titulació: Enginyeria Tècnica Informàtica de Gestió

Director: Jesús Pomar Gomà

Lleida, Novembre 2007

Agraïments.

Al Jesús Pomar, tutor d'aquest TFC, per l'atenció i per l'ajuda en la resolució dels dubtes.

Al Vicente, per compartir els seus coneixements amb mi i per l'ajuda en moments crítics del treball.

A la meva família per tot el suport incondicional rebut durant tota la carrera i per confiar sempre en mi.

Resum.

Aquest Treball Final de Carrera es desenvolupa en el marc d'un projecte de recerca anomenat *Desarrollo y evaluación de un autómata inteligente de alimentación para aumentar la eficiencia alimenticia* realitzat per la Universitat de Lleida (UdL), l'Institut de Recerca i Tecnologia Agroalimentàries (IRTA) i *Agriculture and Agrifood* de Canadà. En aquest treball s'ha realitzat l'anàlisi, disseny i implementació d'un sistema de recepció, filtrat i emmagatzemament de dades integrat en una estructura modular cooperativa de control automàtic distribuït, per automatitzar una explotació porcina.

Els sistemes de subministrament automàtic d'aliment per a porcs d'engreix (alimentació de precisió), generen una gran quantitat de dades que necessiten ésser processades correctament per tal d'extreure'n la informació d'utilitat. Per aquest motiu s'ha creat una Base de Dades on es guardaran i es gestionaran totes les dades recollides pel sistema (*MS Access 2003*). Aquests sistemes d'alimentació automatitzada generen també una sèrie de registres erronis que han de ser depurats per tal de que la informació final sigui la correcta.

Pel desenvolupament s'ha utilitzat l'entorn de programació gràfica *LabVIEW* de *National Instruments*, amb un llenguatge propi de programació *(G)*, així com alguns *toolkits* (*Database*, *Report Generation*) i altres funcionalitats pròpies de *LabVIEW* (*DataSocket*, *WebServer*), degut que tenia que ser integrat amb el *software* que es desenvolupa a l'esmentat projecte de recerca.

També s'ha realitzat el disseny i la implementació d'una eina de consulta a la BD connectada a una altra eina (ambdues en xarxa) capaç de generar informes automàticament de les dades procedents de l'alimentació de porcs d'engreix, amb només un navegador web gràcies a la tecnologia *WebServer* de *LabVIEW*.

El resultat d'aquest projecte ha estat el disseny i la implementació d'un prototip final que consta de diferents mòduls (subprogrames independents entre si, però cooperatius) que permeten l'adquisició, depuració, emmagatzemament, anàlisi, inspecció de dades i la generació de diversos informes predissenyats de manera automàtica; tot això connectat en xarxa.

Frases clau: *LabVIEW*, porcí, generació automàtica d'informes, report, anàlisi de dades, sistema de subministrament automàtic d'aliment, *WebServer, BD, VI.*

Abreviatures.

- **BD**: Base de Dades.
- **DS**: *Data Socket*.
- **DSN**: *Data Source Name* (Nom d'Origen de Dades).
- **DSTP**: *Data Socket Transfer Protocol*.
- **FIRE**: Equip d'enregistrament d'ingestió d'aliment, *Feed Intake Recording Equipment.*
- **FTP**: *File Transfer Protocol*.
- **HTTP**: *HyperText Transfer Protocol*.
- **IP**: Internet Protocol.
- **IRTA**: Institut de Recerca i Tecnologia Agroalimentaria.
- **IVOG**: *Individual Voluntary Feed Intake Recording in Group Housing.*
- **LabVIEW**: Laboratori de banc de treball d'instrument virtual d'enginyeria *Laboratory Virtual Instrument Engineering Workbench*.
- **NI**: *National Instruments*.
- **ODBC**: *Open Data Base Connectivity*.
- **SGBD**: Sistema Gestor de Base de Dades.
- **TCP**: *Transmission Control Protocol*.
- **UDP**: *User Datagram Protocol*.
- **URL**: *Uniform Resource Locator*.
- **VI**: *Virtual Instrument* (programa implementat en LabVIEW).

Índex.

Índex de figures.

1 Introducció

Capítol 1

Introducció

Les noves tecnologies (de la informació) ens ajuden a obtenir una millora de la gestió de la informació, així com una reducció del temps necessari per processar-la. Tot i així, és necessària l'ajuda i participació dels experts en cada una de les matèries a la qual estan destinades les diferents eines informàtiques i tecnològiques, ja que sense la seva col·laboració, aquestes generen molta informació que, un cop processada i obtinguda, es necessari interpretar-la i actuar conseqüentment.

Avui en dia existeixen molts processos, tant industrials, agrícoles, ramaders, etc. que fa uns anys es realitzaven amb mètodes rudimentaris, i que ara depenent totalment de aquests avenços tecnològics.

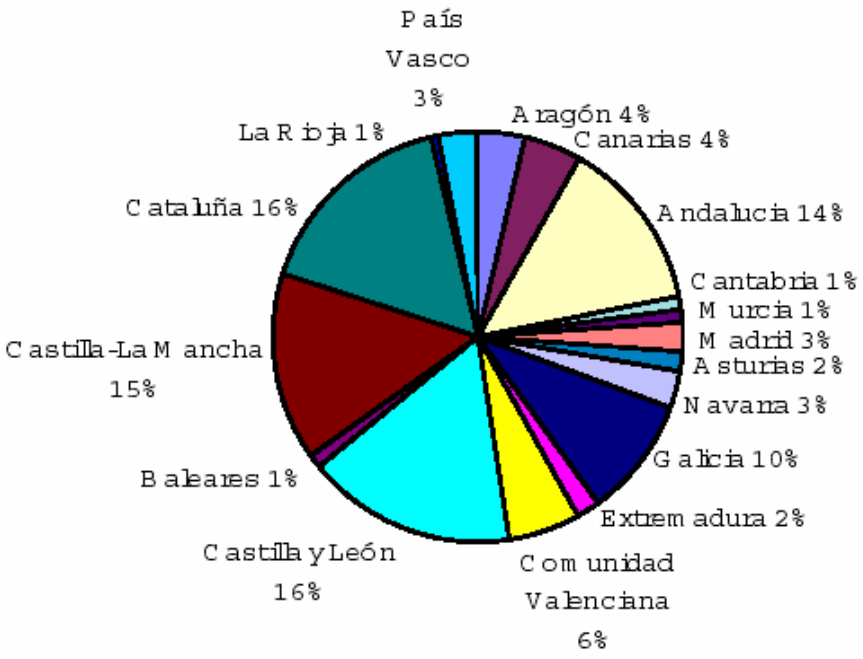


Figura 1.1: Distribució de la ramaderia porcina dins d'Espanya per comunitats.

La importància del sector porcí en la zona justifica per si sola la realització d'aquest projecte, resulta molt convenient poder realitzar un seguiment dels valors d'ingesta del animals, així com de les seves pautes en l'alimentació.

Com podem comprovar a la *figura 1.1,* Catalunya és una de les comunitats autònomes amb més indústria porcina del país, i com veiem a continuació (*figura 1.2*), també té molt pes a la província de Lleida.

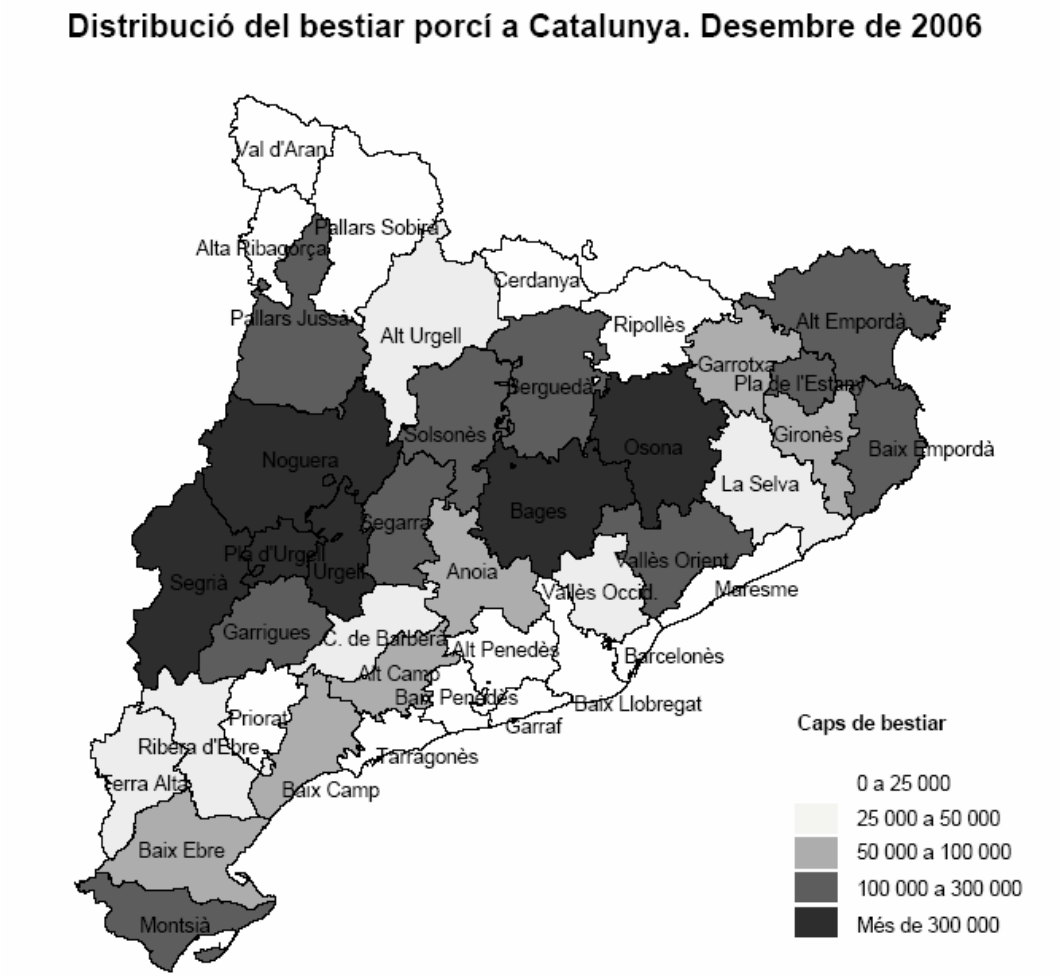


Figura 1.2: Distribució de bestiar porcí a Catalunya.

L'automatització dels sistemes de producció és l'avantguarda de l'era moderna, permeten assolir un nivell competitiu adequat, alhora que milloren i faciliten el maneig dels processos productius.

En el camp de la ramaderia porcina han estat molt pocs els avenços de l'automatització del procés productiu, degut a la seva dificultat i al gran cost que aquests suposaven. Ara, gràcies a la millora tecnològica i a la reducció d'aquests costos es comencen a dissenyar sistemes que permeten l'automatització del maneig d'aquest sector de la ramaderia, permetent un seguiment molt acurat del creixement dels animals, conèixer els hàbits d'alimentació i possibles trastorns i la ràpida detecció d'aquests.

Tots aquests avenços permeten millorar el temps fins a edat de sacrifici, reduint els costos del maneig i permetent al ramader millorar la renda de l'explotació.

1.1. Processament automàtic de les dades per facilitar la presa de decisions.

Durant els últims 20 anys, la informàtica i l'electrònica han aportat molts avenços en el camp de l'adquisició de dades. Tot aquests avenços han facilitat l'adquisició de dades i la capacitat per a generar-les més ràpidament però tots aquests esforços i aquestes millores tecnològiques generen una nova problemàtica: es necessiten eines especialitzades que processin tota aquesta quantitat de dades obtingudes, sense elles es fa difícil extreure'n informació d'utilitat.

L'estudi d'aquesta problemàtica sembla indicar que s'ha avançat suficientment en l'àmbit de l'obtenció automàtica de dades però que per treure profit de tot aquest esforç cal investigar i millorar dins l'entorn del tractament d'aquestes dades, el processament, l'anàlisi i la generació d'informes per tal de poder prendre decisions. Sense l'evolució paral·lela d'aquests dos camps (l'obtenció i el processament de les dades) és gairebé impossible extreure informació útil en un temps raonable.

De fet, sense una eina adequada que automatitzi l'anàlisi i la generació dels informes, la feina s'hauria de fer manualment i comportaria molt temps; d'altra banda, amb l'automatització es podria focalitzar molt més temps i esforç en la presa de decisions i en l'estudi dels resultats enlloc del seu processament.

Aquest treball pretén fer l'anàlisi, disseny i implementació d'una eina informàtica d'ajuda a la recepció, filtrat, emmagatzematge, anàlisi, inspecció de dades i generació d'informes per aquest cas en concret, l'alimentació de porcs d'engreix amb menjadores automàtiques amb control de pes d'aliment IVOG® i, d'aquesta manera, facilitar l'obtenció de resultats finals que ajudin a l'estudi d'aquests animals (*figura 1.3*).

Particularment els sistemes d'alimentació automàtica, generen una gran quantitat de dades diàries que s'han de tractar per tal d'extreure informació de valor. De fet, com s'ha dit anteriorment, aquest gran volum de dades seria poc útil si no se'n fes la seva anàlisi per poder obtenir una informació elaborada. D'aquesta manera, amb les dades obtingudes d'un sistema d'alimentació automàtica, i després del seu tractament, es pot arribar a conèixer quin és el consum diari, setmanal o mensual per lot o per animal i, així mateix, saber quina evolució ha seguit el consum dels animals al llarg del temps, entre molts altres aspectes.

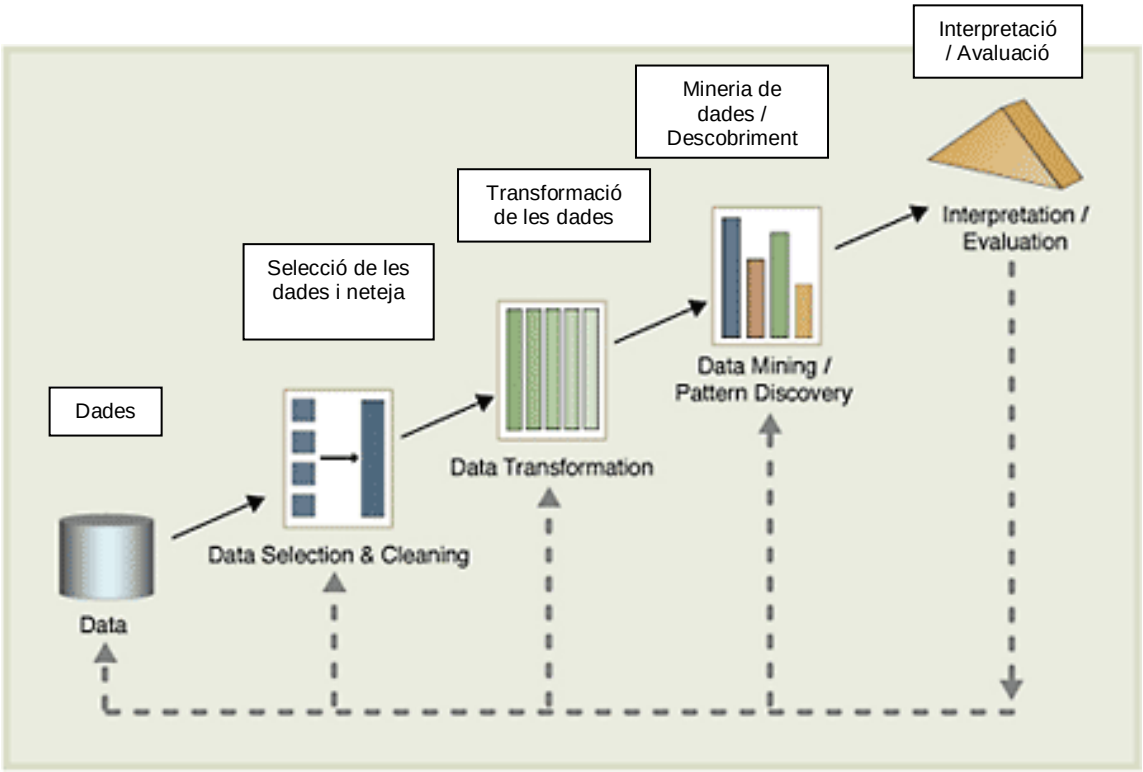


Figura 1.3: Diferents etapes per arribar a l'avaluació i interpretació de les dades.
Font: http://alg.ncsa.uiuc.edu/tools/docs/d2k/manual/dataMining.html

Però per arribar a tenir aquesta informació final cal, primer de tot, seleccionar els registres concrets que ens interessen i fer un tractament previ de les dades ja que per exemple, dins d'aquestes observacions es poden trobar certes incidències degut a que els animals perden l'identificador del cròtal o degut a la presència d'animals rosegadors que creen interferències...I és tal el volum d'informació obtinguda amb aquests sistemes, fins i tot un cop s'han eliminat els errors, que es fa complicat treballar sobre ells si no es té una eina informàtica adequada que permeti automatitzar l'anàlisi i elaborar els informes.

Aquesta informació constitueix la base per a l'anàlisi i presa de decisions i, per tant, és interessant, per a la viabilitat d'un sistema d'alimentació automàtica, disposar d'una eina d'anàlisi que englobi totes les aplicacions indispensables per al tractament de les observacions obtingudes. Una eina que, a més de generar informes, ha de tenir la capacitat d'administrar, visualitzar, analitzar les dades de manera eficient i, sobretot, que accepti algun llenguatge de programació per tal d'automatitzar aquesta feina.

En aquest treball, i amb la finalitat d'assolir els objectius explicats anteriorment, es va partir d'unes dades reals, obtingudes amb un sistema "intel·ligent" d'alimentació per a porcs d'engreix situat al centre *Agriculture and Agrifood* de Canadà, que van ser tractades prèviament per tal d'eliminar les incidències i d'aquesta manera, poder executar correctament la seva anàlisi.

2 Antecedents

Capítol 2

Antecedents

Aquest treball és la continuació i millora en alguns aspectes d'un projecte final de carrera que porta per títol *Anàlisi, disseny i avaluació d'una eina informàtica d'ajuda a l'anàlisi i inspecció de dades d'alimentació de porcs d'engreix, capaç de generar informes automàticament* (N.Riera, 2006).

2.1. Alimentació automatitzada en el sector porcí.

Els sistemes automàtics informatitzats d'enregistrament de consum d'aliment han marcat un abans i un després en l'estudi del comportament alimentari animal degut a la gran quantitat de dades que són capaços de registrar de manera no intrusiva i amb pocs requeriments de mà d'obra. Aquests sistemes estan formats per tres elements bàsics: una menjadora que penja, una bàscula que pesa el seu contingut i una antena que llegeix l'identificador electrònic; a més tenen com a funció el registre, en temps real, dels paràmetres més rellevants, per així fer la posterior avaluació de les dades amb la finalitat d'obtenir la informació d'interès.

Amb aquestes característiques està clar que el desenvolupament d'aquests sistemes, a més de facilitar el control individual del consum d'aliment en animals en grup, obre noves possibilitats d'estudi en l'àmbit productiu, sanitari i comportamental. Des del punt de vista productiu les dades d'aquest sistema han facilitat l'estudi de les corbes de productivitat.

Respecte a la sanitat, amb aquest sistema es pot conèixer a diari els animals que tenen un descens considerable en el seu consum mig diari i, d'aquesta manera, es detecten abans les malalties. De fet, es va observar que amb els subministradors automàtics els porcs que patien grip eren detectats un o dos dies abans que amb l'observació visual. I per últim, aquests sistemes han permès la caracterització de la conducta alimentaria i han facilitat l'estudi dels mecanismes de control del comportament alimentari. Totes aquestes característiques fan que aquest sistemes d'alimentació automàtica tinguin unes bones perspectives de futur.

2.1.1. Sistemes d'alimentació automàtica.

L'aparició de les menjadores automàtiques és producte d'una cadena d'avenços tecnològics posats al servei de l'estudi del comportament alimentari dels animals. Tradicionalment, l'estudi del comportament alimentari es feia per mitjà de l'observació directa i els investigadors anotaven en la seva llibreta els moments i intervals de menjar de cada un dels animals. Més endavant, amb l'ajuda de gravacions en vídeo es va aconseguir alleujar aquesta feina però es continuava necessitant molt de temps i, de totes maneres, no es sabia el pes del menjar ingerit.

L'ús d'un raig de llum amb cèl·lula fotoelèctrica per mesurar la duració de les menjades i l'interval entre elles va suposar un pas endavant. Tot i que amb aquesta tècnica s'evitava l'observació visual encara quedava per saber el pes del menjar ingerit en cada visita.

Alguns intents de valorar el pes de cada ingestió van ser realitzats per Suzuki et al. (1969) i Baldwin et al. (1983), que van connectar la tremuja a una bàscula electrònica. Per altra banda, Savory (1976) va connectar la bàscula a un paper de registre continu per a detectar, sense necessitat d'observació visual, tots els moviments de canvi de pes. No obstant, els registres de paper eren difícils de valorar amb precisió (Fernández, 2001).

Amb el pas del temps, l'ús de microprocessadors connectats a les bàscules va permetre perfeccionar el registre continu dels canvis de pes en la tremuja. Tots aquests avenços

descrits fins ara permetien controlar els animals allotjats individualment, però no en grup. Amb l'aparició del reconeixement electrònic va ser possible identificar la conducta alimentaria individual dels animals allotjats en grup. I és d'aquesta manera quan es van arribar a les estacions automàtiques que s'usen actualment i que es diferencien de les convencionals bàsicament per la protecció que ofereix l'estació alimentadora al porc que està menjant i també pel número limitat d'animals que poden menjar a la vegada. En les estacions alimentadores, comparades amb les convencionals, es dóna un nivell més baix d'agressions durant l'alimentació ja que dues de les parts més vulnerables del porc (les orelles i el cap) queden protegides. Això pot contribuir a incrementar el consum d'aliment, ja que el porc se sent més segur i menys pressionat pels porcs dominants a abandonar l'estació.

Com s'ha comentat anteriorment, les estacions automàtiques dispensadores d'aliment són d'una sola plaça i, per tant, no es pot donar el cas de que dos animals es disposin a menjar a la vegada. És per això que cal fixar correctament el nombre de caps per lot i per tant, per màquina. Les recomanacions pel que fa al nombre d'animals per estació han anat variant al llarg dels anys i segons els autors. Inicialment es partia de 3 a 5 porcs per alimentadora (Englis et al.,1988), mentre que un any després Albal i Graner (1989) recomanaven taxes de 8 a 12 animals. Fins i tot Nielsen et al.(1996) van determinar una taxa de fins a 20 porcs (Fernández, 2001).

Cal tenir en compte que el fet de permetre l'entrada a la menjadora només un animal cada vegada intensifica la competència social ja que aquests animals viuen en estructures jeràrquiques. L'animal dominant del lot anirà les vegades que vulgui a la menjadora quedant nodrit fins a la sacietat mentre que els animals dominats hi podran entrar els temps restant que quedi lliure.

Les estacions alimentadores comercialitzades per a porcs més conegudes són les següents:

- **Individual Voluntary Feed Intake Recording in Group Housing (IVOG®).** És de baixa protecció. Només protegeix el cap de l'animal per a prevenir que dos porcs mengin alhora. Té capacitat per a lots de 8 a 12 animals des dels 30 kg fins als 120 kg. Aquesta estació, a part de funcionar en mode "ad libitum",

també té la possibilitat de proporcionar el menjar de manera restringida. Ha estat utilitzada per grups de recerca holandesos, com per De Haer, per l'Institut de Recerca i Tecnologia Agroalimentària (IRTA) de Monells i per Agriculture and Agrifood Canada de Lennoxville. Aquest equip ha estat l'utilitzat en aquest projecte (*Figura 2.1*).

Figura 2.1: Menjadora automàtica IVOG®.

Font: http://www.insentec.nl

- **Feed Intake Recording Equipment (FIRE®).** És de protecció mitjana. Disposa de diversos tipus de tanques laterals que protegeixen el cos. Utilitzada als grups de recerca escocesos. Té capacitat per a lots de 12 a 15 animals i un error estàndard de l'1% entre el valor real i el mesurat (*Figura 2.2*).

		
Tanca curta: permet una competició moderada protegint el cap i extremitats anteriors. 	**Tanca llarga:** limita més la competició protegint el cos sencer de l'animal. 	**Tanca ACCU-ARM®:** protegeix el cos de l'animal en la seva totalitat i amb unes comportes evita l'entrada de més d'un animal a la vegada. Té un sistema per a pesar els animals i calcular d'aquesta manera l'evolució del pes dels animals, mitjanes diàries, setmanals...

Figura 2.2: Tipus de tanques disponibles per a la menjadora FIRE®.
Font: http://www.osborne-ind.com/lvstcksub/ears/FIRE/options.htm

- **ACEMA 64**: És d'alta protecció. Té una tanca lateral llarga amb porta que tanca després que entri un animal. Utilitzada per investigadors francesos. Té capacitat per a lots de 15 animals amb una precisió de +/- 2 g (*Figura 2.3*).

Figura 2.3**:** Menjadora automàtica ACEMA 64.
Font: http://www.acemo.fr

Un pas més dins de les estacions automàtiques i la tendència actual en aquest àmbit és el d'utilitzar conjuntament un subministrador automàtic de menjar i una bàscula per a mesurar, de manera automàtica, el pes viu de l'animal.

Aquesta bàscula registra el pes de les potes davanteres en cada visita del porc a la menjadora. A partir d'aquest pes, i amb només un error del 2,5 %, es pot conèixer l'evolució del pes viu, dia a dia. Amb aquest avanç s'eviten tots els problemes derivats de pesar manualment: costos de mà d'obra, disturbis provocats per la intervenció de l'home en el moment del pesat i el perill de cometre errors, ja que com va observar Raemekers et al., segons el moment del dia, el pes viu de l'animal varia dins d'un rang del 1,5 % degut al contingut del tracte digestiu i de la bufeta de l'orina (Fernández, 2001).

2.1.2. Adquisició de les dades amb menjadores automàtiques IVOG®

Quan s'estudia la possibilitat d'automatitzar l'anàlisi i inspecció de dades, així com la generació d'informes que es vol obtenir és completament necessari conèixer l'estructura d'aquests arxius i tenir clar com funciona l'equip que obté les dades perquè sense aquesta informació no es disposaria de criteri pel processament d'aquestes dades.

En aquest treball l'obtenció de la informació es va realitzar mitjançant els subministradors automàtics de consum alimentari IVOG® de l'empresa INSENTEC B.V. Aquest subministrador està dotat d'una tremuja que es recolza sobre una bàscula que indica constantment el pes del contingut (*Figura 2.4*).

Aquest subministrador té una capacitat màxima de 30 kg de pinso i està equipada amb unes antenes que llegeixen els codis d'identificació (transponders) que porten els animals en les seves orelles. L'antena del subministrador emet constantment un senyal de radiofreqüència que és captat pels identificadors que porten els porcs a l'orella. Cada vegada que l'identificador d'un animal capta el senyal, s'excita i emet un nou senyal que serà recollit per l'antena.

Figura 2.4**:** Tremuja amb un mecanisme automàtic d'emplenament.
Font: http://www.insentec.nl

El sistema IVOG® és adequat per a l'estudi de patrons alimentaris i consums de porcs individuals en condicions habituals de granja i és per això que organismes de recerca (Centre de Recerca de *Agriculture and Agrifood* Canada de *Lennoxville* i Centre IRTA de Monells), organitzacions de cria (Institute for Pig Genetics d'Holanda) i indústries de pinso de diversos llocs del món utilitzen aquest equip.

Aquest equip registra els consums durant les visites per animal individual i el comportament alimentari dels porcs en grup mantenint de forma natural la seva jerarquització establerta i afavorint d'aquesta manera la interacció entre els animals, factor important per determinar la tendència genètica.
Les característiques de l'equip IVOG® obtingudes de http://www.insentec.nl són:

- És possible subministrar l'alimentació *ad libitum* o de manera restringida.
- Enregistrament dels consums individuals de cada visita.
- Enregistrament opcional dels pesos individuals dels porcs[1].
- Les dades s'emmagatzemen en fitxers diaris pel seu processament posterior.

[1] En aquesta experiència no es va utilitzar.

Cada lot disposa d'un únic subministrador, aquest disseny permet que els porcs siguin desplaçats amb facilitat per altres porcs en el moment de la seva visita a la menjadora. Per evitar que dos animals entrin dins de l'estació al mateix moment i generar d'aquesta manera errors de lectura, l'entrada és ajustable entre 20 i 35 cm, en funció de les dimensions dels porcs (*Figura 2.5*).

Just davant l'entrada a la menjadora es disposa d'una barrera a nivell de terra per evitar que els animals obstrueixin el pas a altres individus en cas de que es quedessin descansant davant l'entrada a l'alimentadora (*Figura 2.6*). Un cop són dins, per evitar pèrdues de pinso i mantenir les instal·lacions més higièniques, els porcs han de colpejar una palanca amb el morro per tal de que es dispensi la dosi d'aliment (*Figura 2.7*).

Figura 2.5: Entrada amb portes ajustables en alçada i amplada.
Font: http://www.insentec.nl

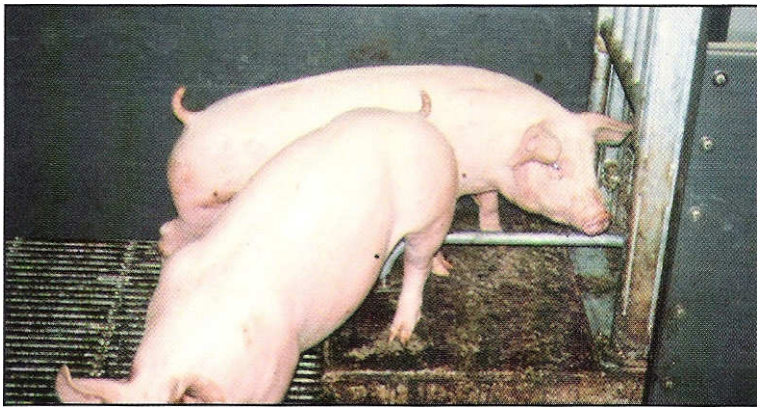


Figura 2.6**:** Barrera a nivell de terra per evitar l'obstrucció de l'entrada per altres porcs. Font: http://www.insentec.nl

Quan un porc entra a l'estació alimentadora, el senyal que emet el transponder del cròtal de l'orella de cada porc és captat i la identificació de l'animal queda registrada juntament amb el moment de la visita i el pes del pinso que conté la menjadora. La mateixa informació es registra electronicament quan el porc abandona l'estació, obtenint-ne la quantitat de pinso consumida per la diferència entre l'entrada i la sortida (*Figura2.8*).

Figura 2.7**:** Dispensador d'aliment per paleta activada amb el morro.Font: http://www.insentec.nl

Figura 2.8**:** Display electrònic del dispensador d'aliment IVOG.

Al final de cada visita, l'estació registra les següents variables:

- **ANIMAL:** identificació de cada porc i associat a un valor d'INSENTEC
- **INSENTEC:** identificador del transponder
- **BOX:** número de lot al que pertany l'animal
- **VALDEB:** pes de la tremuja en kg en entrar l'animal a l'estació
- **VALFIN:** pes de la tremuja en kg en sortir l'animal de l'estació
- **CONSS:** diferència en kg entre els valors de VALFIN i VALDEB, per tant, consum de l'animal
- **TDEB:** hora d'entrada a la menjadora (hh:mm:ss)
- **TFIN:** hora de sortida de la menjadora (hh:mm:ss)
- **Temps:** diferència entre TFIN i TDEB en minuts , per tant, temps de consum
- **DATE:** dia d'entrada a la menjadora (yyyymmdd)

A la següent taula es mostra un fragment dels fitxers de tipus text que es creen.

ANIMAL	INSENTEC	BOX	VALDEB	VALFIN	CONSS	TDEB	TFIN	Temps	DATE
115	18063143	1	14,01	13,78	0,23	02:12:44	02:16:05	3,21	20040127
FILLING	0	1	13,78	14,33	-0,55	02:16:05	02:16:07	0,02	20040127
115	18063143	1	14,33	14,3	0,03	02:16:08	02:16:29	0,21	20040127
102	18048216	1	14,31	14,13	0,18	02:23:09	02:25:57	2,48	20040127
102	18048216	1	14,13	14,14	0,01	02:26:02	02:26:04	0,02	20040127
116	18048145	1	14,12	14,07	0,05	04:06:04	04:06:42	0,38	20040127
105	18046811	1	14,08	13,97	0,11	04:41:41	04:43:26	1,45	20040127
FILLING	0	1	13,97	14,52	-0,55	04:43:27	04:43:27	0	20040127
105	18046811	1	14,52	14,47	0,05	04:43:28	04:44:37	1,09	20040127
105	18046811	1	14,49	14,49	0	04:44:54	04:46:40	1,46	20040127
105	18046811	1	14,49	14,26	0,23	04:46:40	04:48:00	1,2	20040127
105	18046811	1	14,26	14,17	0,09	04:48:03	04:49:16	1,13	20040127
108	18047969	1	14,17	14,09	0,08	04:50:20	04:51:55	1,35	20040127
108	18047969	1	14,09	14,01	0,08	04:51:56	04:53:49	1,53	20040127
108	18047969	1	14,01	13,54	0,47	04:53:54	05:02:54	9	20040127
FILLING	0	1	13,54	14,09	-0,55	05:02:56	05:02:56	0	20040127
108	18047969	1	14,09	14,01	0,08	05:03:01	05:04:37	1,36	20040127
FILLING	0	1	14	14,54	-0,54	05:14:42	05:14:44	0,02	20040127
105	18046811	1	14,54	14,4	0,14	05:15:28	05:18:14	2,46	20040127
105	18046811	1	14,4	14,36	0,04	05:18:17	05:18:59	0,42	20040127
102	18048216	1	14,36	14,32	0,04	05:20:42	05:22:00	1,18	20040127
103	18055929	1	14,32	14,24	0,08	05:24:22	05:25:28	1,06	20040127
143	18047887	1	14,09	14,09	0	05:57:16	05:57:18	0,02	20040127
.									
.									
.									

Figura 2.9**:** Fragment d'un fitxer de tipus text IVOG®.

La màquina, per defecte, guardarà les dades obtingudes pels sensors en fitxers *.txt* diaris, la qual cosa té la seva utilitat, però també té les seves inconveniències com fer una cerca d'un animal concret o de tot un lot d'animals pot trigar bastant temps. Això és un dels objectius de millora d'aquest treball, fer un sistema que guardi i gestioni la informació en bases de dades per obtenir més eficiència del sistema global.

Més endavant, a l'anàlisi de la base de dades, es canviaran el nom d'algunes variables de la informació rebuda i s'afegiran més també, ja que es portarà a terme un procés de filtratge de la informació abans d'emmagatzemar-la.

2.2. Arquitectura modular en sistemes de control.

L'ús de la instrumentació virtual per implementar un sistema de control és un dels factors clau per garantir l'èxit en aquest treball. Igualment resulta molt important definir o determinar una correcta arquitectura modular del sistema de control, es a dir, crear un disseny conceptual previ i una estructura operacional de cada mòdul del sistema. En aquest sentit l'arquitectura modular és una tipologia específica d'un sistema distribuït.

2.2.1. Sistemes distribuïts.

No existeix un consens en la bibliografia sobre la definició d'un sistema distribuït. Altrament, el que si és acceptat per tothom són dos aspectes que ho caracteritzen:

- Està format per un conjunt de màquines autònomes connectades entre sí de manera que els usuaris treballen amb el sistema com si fos una sola màquina.
- El sorgiment dels sistemes distribuïts fou possible gràcies a la invenció de les xarxes d'ordinadors juntament amb el desenvolupament tecnològic d'enllaços de comunicació d'alta velocitat. Tot això va permetre la construcció de les xarxes del tipus LAN i WAN.

Dins de les principals motivacions que comporta a la construcció d'un sistema distribuït es troba: la economia, la velocitat, la distribució inherent, la confiabilitat i la escalabilitat.

Aquestes motivacions sorgeixen d'un anàlisi comparatiu entre la utilització de sistemes centralitzats en vers la utilització de sistemes distribuïts. Un sistema centralitzat s'entén com un sistema que posseeix una sola CPU amb els seus respectius recursos, a la qual es poden connectar terminals.

En aquesta definició cal destacar dos aspectes. Un, el hardware. La definició parla de màquines autònomes, es a dir, que poden operar sense la supervisió de cap altra. Dos, el software, que ha d'aconseguir que els usuaris del sistema ho vegin com una màquina central convencional i única.

Malgrat les avantatges descrites anteriorment, els sistemes distribuïts també tenen les seves debilitats. La primera és com ha de ser el software dels sistemes distribuïts en les àrees de disseny, implementació, funcionalitat, ús, etc. No es coneixen amb precisió com han de ser els sistemes operatius, llenguatges o aplicacions d'aquests sistemes.

El segon problema associat als sistemes distribuïts és la seva total dependència de la xarxa de comunicació. Quan la distribució creix, el tràfic en la xarxa creix també i pot arribar a saturar-se. És llavors quan resulta necessari reemplaçar la instal·lació de xarxa. S'haurà de realitzar una costosa operació de recablejat de l'edifici, reemplaçar les targes de xarxa, etc. D'altra banda, un tall en la xarxa pot deixar al sistema completament inutilitzat degut a la seva total dependència a ella.

Un tercer problema que poden tenir els sistemes distribuïts és la seva falta de seguretat. Un dels avantatges dels sistemes distribuïts és la possibilitat de compartir dades ja que proporcionen un fàcil accés als mateixos. Aquesta accessibilitat resulta molt perillosa quan les dades han de ser privades.

3 Objectius

Capítol 3

Objectius

Els objectius del treball són el desenvolupament d'una aplicació informàtica especialitzada, d'una banda, en l'**adquisició** i la **gestió** de dades (en xarxa) generades en una explotació porcina. Aquestes dades provindran d'un sistema d'alimentació automàtic, el qual genera moltes dades però fins ara no es gestionaven d'una manera adequada amb bases de dades, sinó que es feien amb fitxers de text (*.txt*), un per dia. Per realitzar aquesta tasca s'haurà d'observar i estudiar quins tipus de dades s'han de gestionar, per poder dissenyar una base de dades d'acord a elles.

D'altra banda, l'eina desenvolupada en aquest treball, incorporarà la opció de **generació automàtica d'informes** (via *Internet*) per tal de ser integrada, en el desenvolupament d'un sistema "intel·ligent" d'alimentació de porcs d'engreix; aquest projecte de més envergadura el desenvolupa la Universitat de Lleida (UdL), l'Institut de Recerca i Tecnologia Agroalimentàries (IRTA) i *Agriculture and Agrifood* de Canadà.

Les funcions principals de l'eina informàtica seran la inspecció i el filtrat de les dades en brut, la seva inserció a la base de dades (creada i dissenyada prèviament), la consulta a la BD via *Internet*, així com la posterior generació de diversos informes automàtics, tant generals com personalitzats a partir d'una consulta prèvia efectuada per l'usuari, tant local com remotament.

Tot aquest desenvolupament s'implementarà en un entorn de programació gràfica anomenat *LabVIEW* de l'empresa *National Instruments*, el qual té un llenguatge de programació propi, anomenat *G*, molt potent, amb multitud d'eines molt específiques orientat a la implementació d'instrumentació virtual.

Com a treball implícit del projecte es farà una valoració de l'eina principal utilitzada (*LabVIEW 7.1* i les seves *toolkits* o llibreries específiques) per determinar si és l'elecció més adequada i quines alternatives podria tenir.

4 Eines software utilitzades

Capítol 4

Eines software utilitzades.

4.1. *LabVIEW 7.1.* Justificació per requeriments del projecte.

Aquest treball de fi de carrera s'emmarca, com s'ha dit a la introducció, dins d'una part d'un projecte de gran envergadura, l'objectiu del qual és aconseguir automatitzar i controlar tots els processos que tenen lloc en una granja d'engreix d'animals, concretament una granja porcina. És per això que els requeriments d'aquest treball, es determinen en gran part per aquest motiu.

Un dels requeriments més importants d'aquest treball era l'eina utilitzada per implementar el prototip, que és *LabVIEW 7.1,* amb la qual es venia treballant en aquest projecte en la programació de les diferents parts.

LabVIEW és una eina gràfica de test, control y disseny mitjançant la programació. El llenguatge que utilitza s'anomena *llenguatge G.*

La **programació *G*** és el cor de *LabVIEW*, i es diferencia d'altres llenguatges de programació com *C* o *Basic*, en que aquestos estan basats en text, mentre que *G* és una programació gràfica.

Aquest programa va ser creat per la empresa nord-americana *National Instruments* al 1976, per funcionar sobre màquines MAC i va sortir al mercat per primera vegada al 1986. A dia d'avui està disponible per a les plataformes Windows, UNIX, MAC i Linux.

Laboratory Virtual Engineering Workbench (LabVIEW), és un entorn de programació de fluxe de dades mitjançant el seu propi llenguatge gràfic anomenat *G.*

Aquest llenguatge fou dissenyat per facilitar el desenvolupament de sistemes d'adquisició de dades, anàlisi, monitorització i control d'aplicacions (*figura 4.1*), tant per científics com per enginyers, que necessiten un software per interactuar amb equip de laboratori.

L'us d'un llenguatge de programació gràfica per implementar un programa de simulació promou habilitats cognitives específiques [Faraco *et al*. 2006]:

- L'habilitat de formular i construir un model d'un concepte específic que descriu un fenomen d'on el problema és analitzat i descompost en unitats simples.
- L'habilitat de transformar aquest model en una solució lògica implementant-ho en un programa.
- L'habilitat d'entendre i utilitzar objectes de *LabVIEW* , llegir i manipular les dades d'entrada al programa, conèixer la modificació de les dades i respectar un ordre tant en la forma com en el procediment de creació.
- L'habilitat de provar el propi programa (instrument virtual) corregir errors i manipular-lo per fer-lo evolucionar a una altra versió.

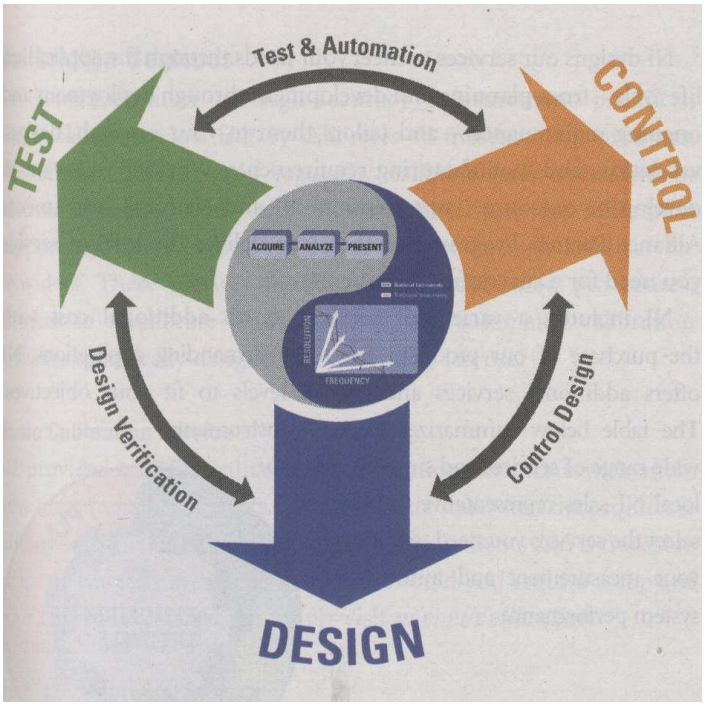


Figura 4.1: Us de la instrumentació virtual en la enginyeria desenvolupant aplicacions de: test, control i disseny.

Principals usos:

LabVIEW es utilitzat principalment per enginyers y científics per tasques com:

- Adquisició de dades.
- Control d'instruments.
- Automatització industrial o PAC (Controlador de Automatització Programable).
- Disseny de control: prototipatge ràpid y hardware-en-el-bucle (HIL).
- Disseny embebit.
- Domòtica.
- ...

LabVIEW es pot connectar de manera transparent con tot tipus de hardware incloent instruments de escriptori, targetes insertables, controladors de moviment i controladors lògics programables (*PLC*s).

4.1.1. Instrumentació virtual.

Els programes implementats amb *LabVIEW* s'anomenen ***VI*** (*Virtual Instrument*), el que dona una idea del seu us en origen: el control de instruments. Aquests *VI*'s estan dividits en dues parts diferenciades: la interfície (panel frontal) i el codi (diagrama de blocs).

L'objectiu principal de la instrumentació virtual és substituir i ampliar elements *hardware* per altres *software* mitjançant un processador (normalment un PC).
Executant un programa específic, aquest es comunica amb els dispositius per configurar-los i llegir les mesures o les dades rebudes.

Algunes de les avantatges més importants de la instrumentació virtual són:

- És capaç d'automatitzar les mesures.
- Capacitat de processament de la informació.
- Visualització i actuació remota...

Un instrument virtual consisteix en un computador de tipus industrial, o bé una estació de treball (potser un PC) equipada amb un potent *software* i un *hardware* econòmic que compleixen, en conjunt, les funcions dels instruments tradicionals, tant d'automatització com de control de dades.

Els *VI* representen un distanciament fonamental dels sistemes de instrumentació basats en el *hardware*, a sistemes centrats en el software els quals aprofiten la potencia de càlcul, productivitat, exhibició i capacitat de connexió dels ordenadors personals i estacions de treball. Encara que el PC i la tecnologia de circuits integrats han experimentat avenços significatius en les últimes dècades, és el software el que realment proveeix l'avantatge de construir sobre aquesta potent base de *hardware*, innovant i reduint costos.

Instruments autònoms tradicionals, tals com oscil·loscopis i generadors de ones, són molt poderosos, però també són molt cars i dissenyats per una tasca específica definida pel fabricant. La qual cosa fa que l'usuari no pugui expandir o personalitzar més tasques. A més a més s'ha de desenvolupar una tecnologia especial amb components molt costosos per construir-lo.
Degut a que els *VI*, estan basats en el PC, aquests aprofiten els beneficis de les últimes tecnologies dels ordenadors personals. Aquests avenços en tecnologia i rendiment inclouen processadors i diversos sistemes operatius (*Microsoft Windows, MAC OS, Linux*...), així com accés a *Internet* i altres eines.

A excepció dels components especialitzats, l'arquitectura general dels instruments tradicionals és molt similar a la d'un instrument virtual. Els dos requereixen d'un o més microprocessadors, ports de comunicació i capacitat de mostrar resultats, d'adquirir dades. La principal diferencia entre un i l'altre és la flexibilitat i el fet de poder modificar i adaptar l'instrument a les necessitats particulars de l'usuari. Un *VI* té la possibilitat d'ampliar les seves funcionalitats mitjançant un software, d'una manera relativament fàcil i econòmica, i estarà limitat únicament per la potencia (i el preu) d'aquest software.

Una de les principals avantatges d'aquest software és la seva modularitat. Davant d'un gran projecte els enginyers generalment aborden la tasca dividint-la en unitats funcionals. Aquestes tasques subsidiàries són més fàcil de manipular i més fàcils de provar degut a les menors dependències al *hardware* i que podrien causar comportaments inesperats i/o inadequats.

Un altre avantatge de la instrumentació virtual són les aplicacions distribuïdes. Aprofitant el creixent desenvolupament en la tecnologia de xarxa i d'*Internet*, és més comú utilitzar la potencia de connectivitat dels instruments virtuals amb el fi de compartir tasques i delegar-les en diferents *VIs*, afavorint així a la especialització del software, perque cada instrument farà una tasca molt específica, i també afavorint la possible ampliació o evolució del sistema, ja que no cal retocar tot el codi, sinó només la part especifica que realitzarà una tasca concreta, respectant doncs els principis de la enginyeria del software.

4.1.1.1. Programació gràfica en *LabVIEW*: (llenguatge *G*).

El llenguatge *G*, és el llenguatge propi de l'entorn de programació *LabVIEW* i es basa en la programació gràfica, es a dir, alhora d'implementar codi no l'escrivim, sinó que el "dibuixem" mitjançant una sèrie de funcions i de llibreries pròpies d'un llenguatge de programació.

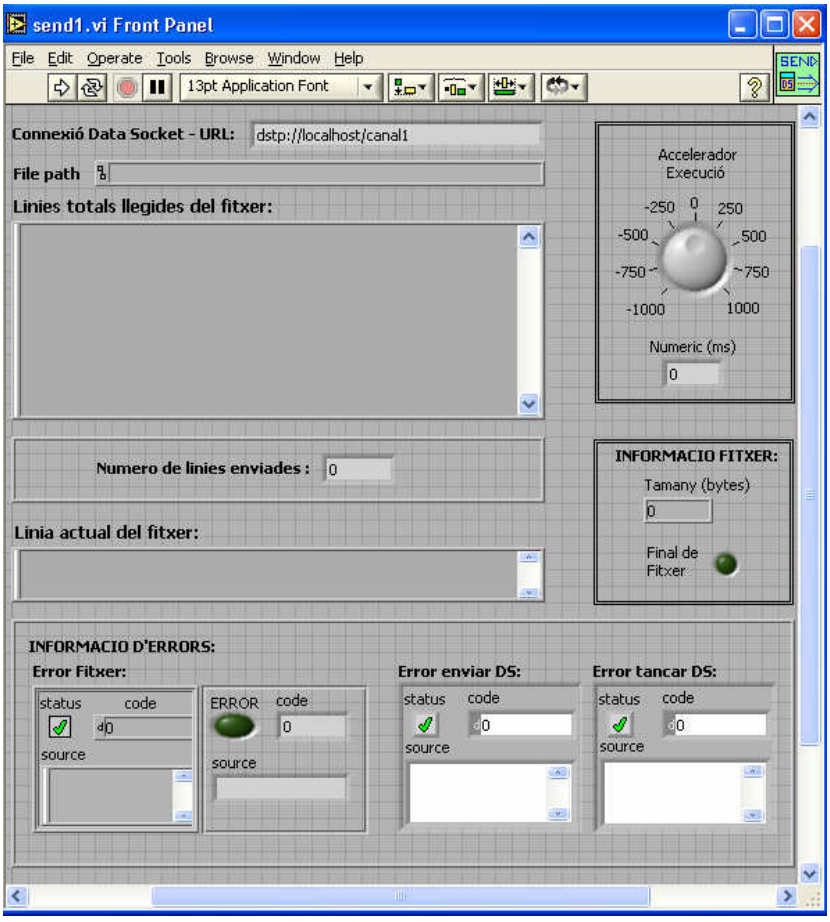


Figura 4.2-a: Panel frontal d'un programa implementat en *LabVIEW 7.1*.

En el panel frontal es defineixen els *controls* i els *indicadors* que es mostren per pantalla. El *Diagrama de Blocs* és el programa pròpiament dit, on es defineixen les seves funcionalitats implementades en *llenguatge G*.

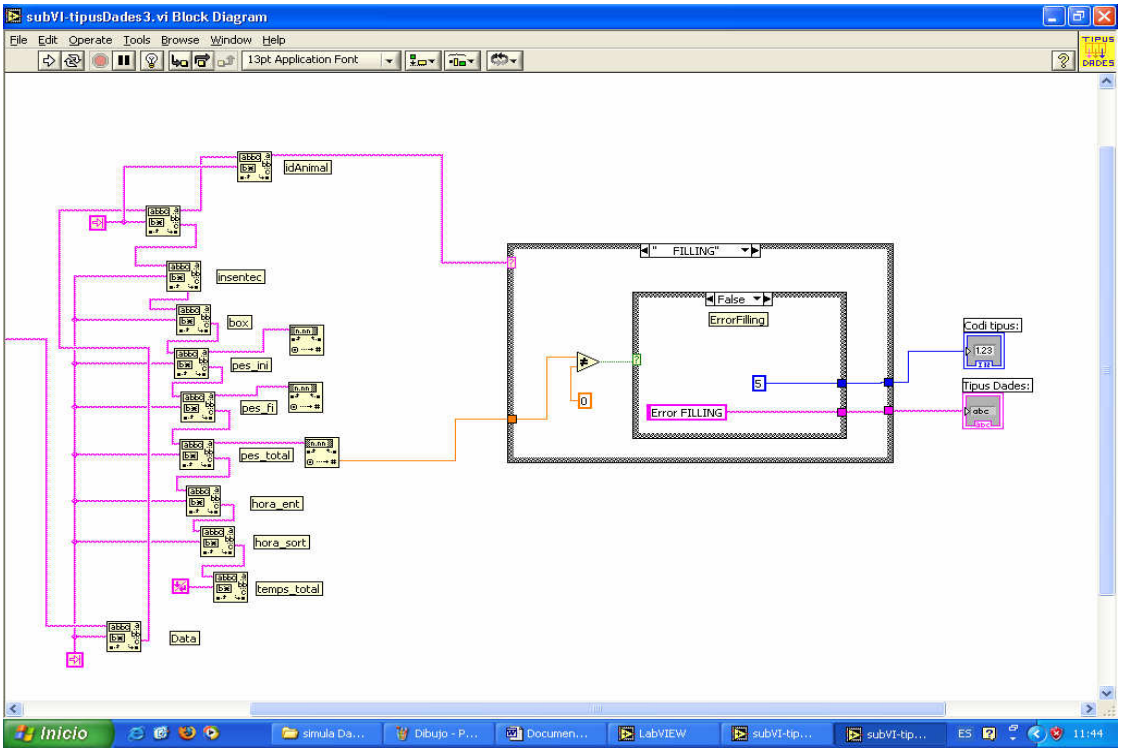


Figura 4.2-b: Diagrama de blocs d'un programa implementat en *LabVIEW 7.1*.

Figura 4.3-a: Funcions de *LabVIEW 7.1*.

Les funcions del llenguatge *G* de *LabVIEW* es despleguen quan clickem amb el botó dret del ratolí dins del diagrama de blocs del programa que estem implementant. Les funcions comuns, i algunes específiques del llenguatge que es van utilitzar en TFC són les següents:

1. *Structures.*
2. *Numeric.*
3. *Boolean.*
4. *String.*
5. *Array.*
6. *Cluster.*
7. *Comparison.*
8. *Time & Dialog.*
9. *File I/O.*
10. *NI Measurements.*
11. *Waveform.*
12. *Analyze.*
13. *Instrument I/O.*
14. *Application control.*
15. *Graphics & Sounds.*
16. *Communication.*
17. *Report Generation.*
18. *Advanced.*
19. *Select a VI...*
20. *Decorations.*
21. *User Libraries.*
22. *Database.*
23. *DIAdem.*

Figura 4.3-b*:* Funcions *LabVIEW* numerades.

Els *VIs* son **jeràrquics i modulars**. Poden utilitzar-se com programes d'alt nivell o com subprogrames o subrutines d'altres programes o subprogrames. Quan un *VI* s'utilitza dins d'un altre *VI*, es denomina *subVI*. La icona i els connectors d'un *VI* funcionen com una llista de paràmetres gràfics de forma que altres *VIs* poden passar dades a un determinat *subVI*.

4.1.2. *Toolkits* utilitzats en la implementació de l'eina.

En l'entorn *LabVIEW* , existeixen moltes maneres de dissenyar prototips de software, i cada tipus de prototip necessita unes llibreries específiques i opcionals, anomenades *toolkits (figura 4.4)*, que no estan en l'entorn del programa "base". És per aquest motiu que necessitem instal·lar algunes d'aquestes llibreries, que no són més que conjunts d'eines o funcions fetes per implementar diferents tipus de programes en l'entorn *LabVIEW* i que no venen instal·lades per defecte en la nostra versió del programa *LabVIEW 7.1*.

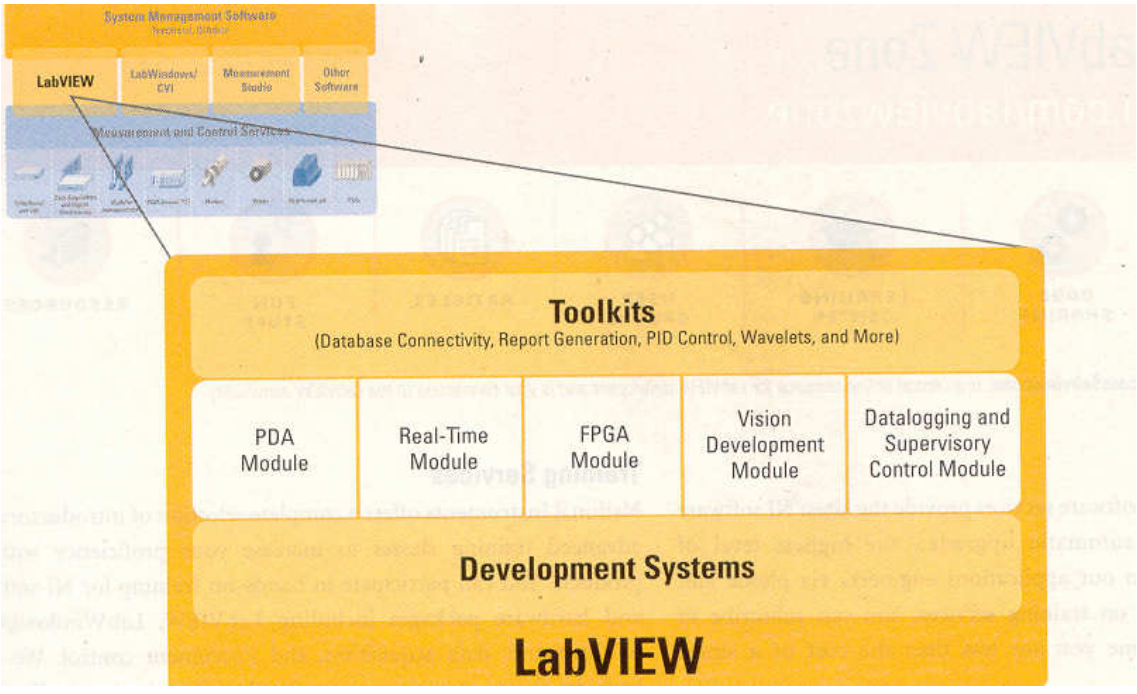


Figura 4.4: Alguns *toolkits* específics associats a *LabVIEW 7.1*.

Aquesta versió consta d'una col·lecció de *CDs* dels quals s'utilitzen dos en aquest treball: *development* i *toolkit*.

4.1.2.1. *DataBase.*

Aquest *toolkit* serveix per poder implementar prototips amb connexions a bases de dades, per poder realitzar consultes, insercions i d'altres funcionalitats pròpies de les *BDs* i del llenguatge *SQL*.

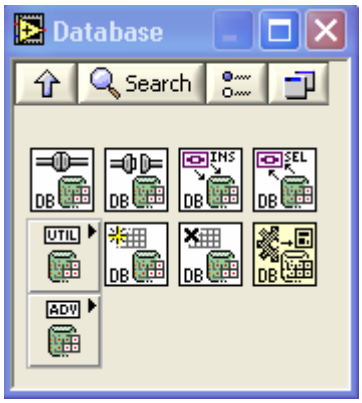


Figura 4.5-a: Funcions especifiques de Bases de Dades.

El *LabVIEW Database Connectivity Toolkit* és un conjunt d'eines, o funcions definides per treballar en l'entorn de programació de *LabVIEW* mitjançant el llenguatge *G*. A través d'aquest *toolkit* podem comunicar-nos amb bases de dades creades amb diferents gestors de *BD* , tant localment com de manera remota. També ofereix la possibilitat de fer consultes, insercions o altres operacions pròpies de bases de dades, sense tenir que escriure ni conèixer el llenguatge *SQL*, només utilitzant les funcions predefinides del *toolkit*. Si per el contrari el que volem és fer consultes avançades i de caire més complexe, també podem escriure les consultes directament en *SQL*.

A la *figura 4.5-a* podem veure les funcions propies d'aquest toolkit, així com les funcions avançades (*figura 4.5-b*), que es despleguen quan clickem a la última casella.

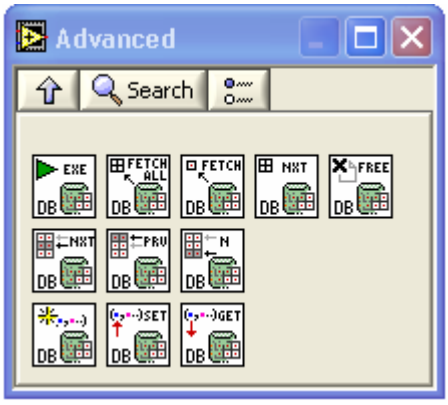


Figura 4.5-b: Funcions avançades del *Database toolkit*.

Amb aquest *toolkit*, podem connectar-nos a la majoria de bases de dades:

- Microsoft Access
- Microsoft SQL Server
- Oracle
- Visual FoxPro
- dBase
- Paradox

El *Database Connectivity Toolkit* és la evolució directa del *SQL Toolkit*, però amb noves capacitats i millores considerables, com per exemple:

- Insertar i seleccionar dades d'una *BD*.
- Crear i esborrar taules.
- Llistar taules i columnes en una *BD*.
- Acceptar múltiples operacions (transaccions).
- Executar operacions immediates en *SQL*.
- Seleccionar informacions d'una *BD* i guardar-la en un fitxer *XML*.

Amb aquest toolkit ens podem connectar a diferents tipus de BD utilitzant els diferents tipus de connexions: *ADO-compliant OLE DB provider* o *ODBC driver*.

El *Database Connectivity Tool*kit és compatible amb versions *LabVIEW 6.x* i superiors. Quan utilitzem aquest *toolkit* com una part d'una *distribution package* més gran, no és necessari que tinguem instal·lat el *runtime* per utilitzar-lo.
El *toolkit* és compatible amb les següents plataformes: *Windows 2000/NT/Me/XP*.

4.1.2.2. *Report Generation Toolkit (for Microsoft Office).*

Aquest *toolkit* serveix per poder implementar prototips, els quals, puguin realitzar informes o *reports*, enfocats al paquet de programes *MS Office*. En aquest treball concretament es realitzen amb el programa *MS Excel*.

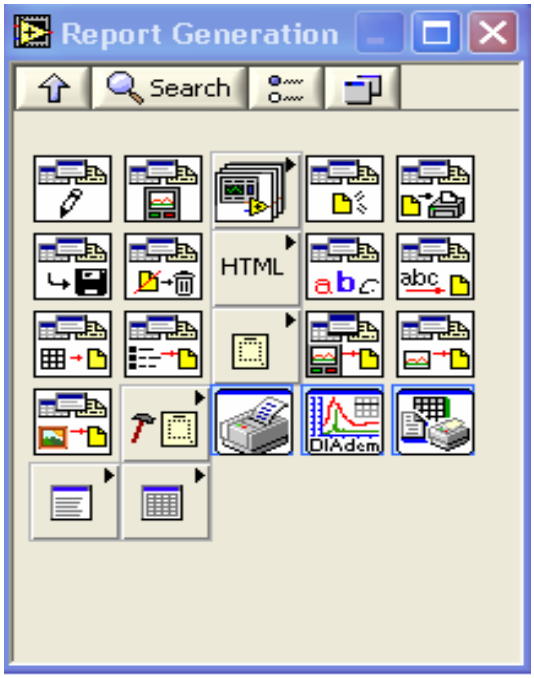


Figura 4.6-a: Funcions específiques del *toolkit Report Generation*.

Com podem veure en la *figura 4.6-a* hi ha bastants funcions pròpies del *Report Generation Toolkit*, ja que aquesta eina ens permet fer varis tipus de *reports*, com per exemple *HTML*. Però en aquest treball ens centrarem en una part especifica d'aquest *toolkit*, com és *Report Generation for MS Office*, concretament en fer reports utilitzant *MS Excel*.

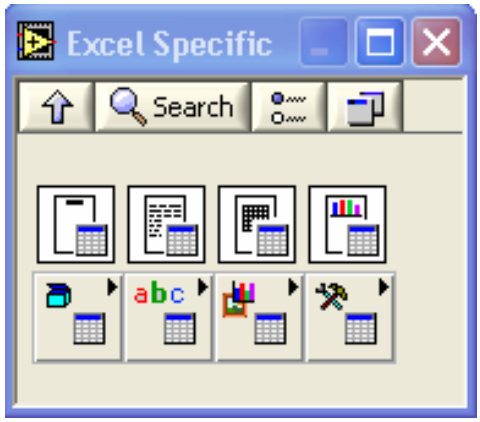


Figura 4.6-b: Funcions específiques de *MS Excel* del *Report Generation*.

4.1.3. Altres eines de *LabVIEW*.

En l'entorn de *LabVIEW* existeixen múltiples eines integrades, les quals es poden utilitzar sense instal·lar cap *toolkit*. Les utilitzades en aquest treball són: el *DataSocket* i el *WebServer*. A continuació explicarem que són, per a que serveixen i com s'han utilitzat en el treball en qüestió.

4.1.3.1. *WebServer.*

Un servidor web és un programa que implementa el *protocol HTTP* (*hypertext transfer protocol*), no l'hem de confondre amb el servidor web que també se li anomena a la màquina que fa de servidor físic. Aquest protocol està dissenyat per transferir el que diem hipertexts, pàgines web o pàgines *HTML* (*hypertext markup language*): texts complexos amb enllaços, figures, formularis, botons y objectes incrustats com animacions, etc.

La funció que realitza un servidor web és de mantenir-se a l'espera de *peticions HTTP* efectuades per un client *HTTP* que solem conèixer com *navegador*. El navegador realitza una petició al servidor i aquest li respon amb el contingut que el client solicita. A mode d'exemple, quan escrivim *www.udl.es* en el nostre navegador, aquest realitza una petició *HTTP* al servidor d'adreça. El servidor respon al client enviant el codi *HTML* de la pàgina; el client, una cop ha rebut el codi, l'interpreta i el mostra per pantalla. Com veiem amb aquest exemple, el client és l'encarregat d'interpretar el codi *HTML*, es a dir, de mostrar les fonts, els colors i la disposició dels textos i objectes de la pàgina; el servidor tan sols es limita a transferir el codi de la pàgina sense portar a terme cap interpretació de la mateixa.

Sobre el servei web *clàssic* podem disposar de aplicacions web. Aquestes són fragments de codi que s'executen quan es realitzen certes peticions o respostes *HTTP*. S'ha de distingir entre:

- Aplicacions en el costat del client. El client web es l'encarregat d'executar-les en la màquina del usuari. Són les aplicacions tipo *Java* o *Javascript*: el servidor proporciona el codi de les aplicacions al client i aquest, mitjançant el navegador, les executa. Es necessari, per tant, que el client disposi d'un navegador amb

capacitat per executar aplicacions (també s'anomenen *scripts*). Normalment, els navegadors permeten executar aplicacions escrites en llenguatge *javascript* i *java*, encara que poden afegir-se més llenguatges mitjançant l'us de *plugin*s.

- Aplicacions en el costat del servidor. El servidor web executa l'aplicació; aquesta, un cop executada, genera cert codi *HTML*; el servidor pren aquest codi acabat de crear i l'envia al client mitjançant el protocol *HTTP*.

Les aplicacions de servidor solen ser la opció per la que s'opta en la majoria de les ocasions per realitzar aplicacions web. La raó és que, al executar-se aquesta en el servidor i no en la màquina del client, aquest no necessita cap capacitat addicional, com sí succeeix en el cas de voler executar aplicacions *javascript* o *java*. Així doncs, qualsevol client dotat d'un navegador web bàsic pot utilitzar aquest tipus d'aplicacions.

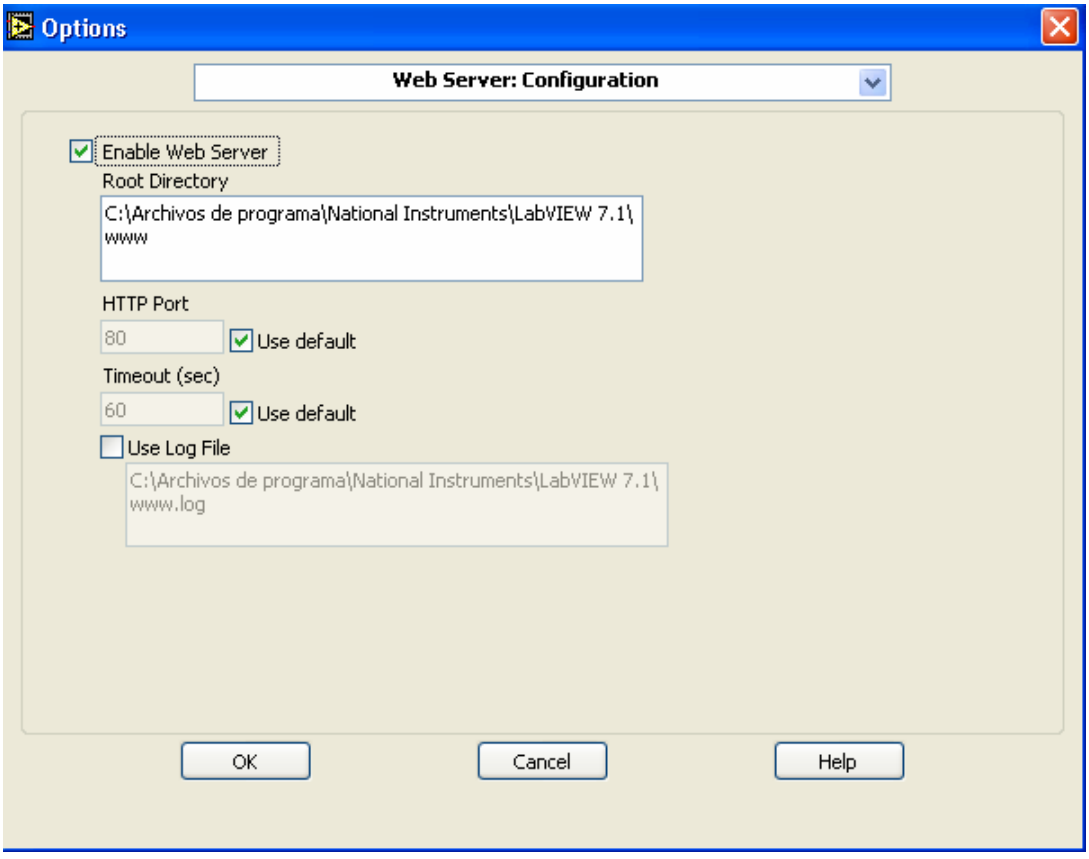


Figura 4.7: Pantalla de configuració del servidor web de *LabVIEW 7.1*.

Per configurar el *webserver* de *LabVIEW* hem de clickar a la barra de menús:

Tools ➔ Options ➔ Web Server: Configuration.

Un cop a la pantalla de configuració (*figura 4.7*) s'ha de triar el directori on estaran guardades les nostres pàgines web de *LabVIEW* , el port *HTTP* que vulguem, el temps màxim de carrega en segons.

Per editar les pàgines web de *LabVIEW* s'ha utilitzat l'eina integrada a *LabVIEW*: *web publishing tool (figura 4.8)*, la qual transforma els *VIs* seleccionats en pàgines web.

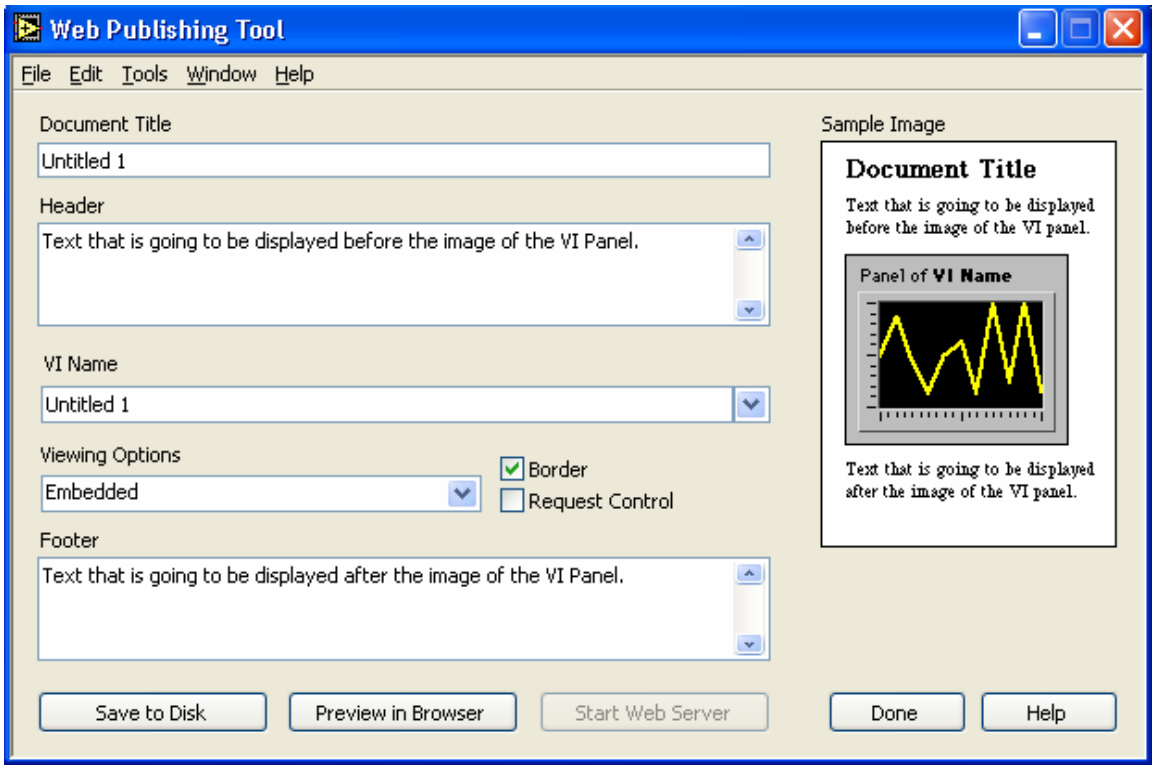


Figura 4.8: Editor de pàgines web de *LabVIEW*, *web publisher*.

4.1.3.2. *DataSocket.*

El *DataSocket Transfer Protocol* és una eina de comunicació basada en els protocols *TCP/IP*, la qual simplifica l'intercanvi de dades entre diferents aplicacions d'un ordenador o entre més d'un connectats per xarxa.

DataSocket va ser desenvolupat específicament per publicar dades que s'actualitzen contínuament (*live data)*, o dades que s'actualitzen continuament en una xarxa. Anteriorment existien diferents tipus de hardware, de software i de protocols implicats en les aplicacions. Aquesta complexitat forçava al programador a negociar entre els diversos protocols para la transferència de dades, que requeria temps i recursos.

Com solució a això, *National Instruments* va desenvolupar *DataSocket*, el qual permet la transferència de dades sobre diversos protocols (*DSTP, OPC, LOOKOUT, HTTP, FTP,* y *local file access*).

En aquest treball en centrarem en el *DataSocket Transfer Protocol (DSTP)* que és el que em fet anar per implementar una simulació de comunicació entre dos programes o *VIs*. La via de comunicació d'aquest protocol és que utilitza el servidor de *DataSocket* (*DS Server*). Aquesta arquitectura de la comunicació es basa en tres parts principals: un editor de los dades, un servidor *DataSocket* y un client de *DataSocket*.

DataSocket simplifica aquest procés proporcionant una sèrie de funciones específiques de crides simples per transferir dades i una arquitectura fàcil de utilitzar del servidor-client per compartir dades.

El editor de dades, pot ser un programa que reculli dades d'algun sensor, per després publicar al *DataSocket Server*.

El servidor de *DataSocket* es una aplicació que pot funcionar en la mateixa màquina o podria ser un ordinador remot. Aquest servidor de *DataSocket* reservarà una quantitat petita de memòria i permetrà que les dades siguin emmagatzemades en aquest servidor. Un cop que les dades són transferides al servidor, aquestes són accessibles des de qualsevol màquina client fàcilment.

L'adreça del *DataSocket* és similar a la *URL* que normalment s'utilitza per pàgines web. La primera part identifica el nom de l'ordinador o direcció *IP*, mentre que la última part identifica el canal que es vol llegir o escriure.

Per exemple: dstp://192.168.0.2/canal1, si volem escriure l'adreça *IP* de la màquina

O també: dstp://nom/canal2, en el cas que vulguem escriure el nom del *PC*.

On ***dstp*** indica el protocol que utilitzem (*Data Socket Transfer Protocol*).

DataSocket Server.

És el servidor que connecta totes les aplicacions *DS* i permet la comunicació entre ells (*figura 4.9*).

Aquest servidor està escoltant pel port 3015 les peticions dels clients, com un servidor web. Un cop arriba la petició d'un client, el *DS Server* comprova si el client té permí per accedir al servidor, si el client té permís es processa la petició.

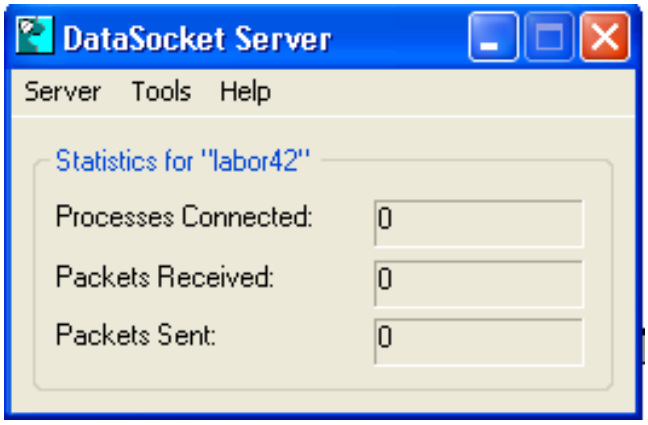


Figura 4.9: *DataSocket Server*, programa servidor per comunicar-se mitjançant *DSTP*.

El *DS Server* sempre ha de estar funcionant per aquells clients que vulguin realitzar operacions de lectura i/o escriptura.

4.1.3.3. *RunTime de LabVIEW.*

Per poder utilitzar un programa implementat en *LabVIEW* no cal tenir instal·lat tot l'entorn de programació de *LabVIEW*, amb tot el que això suposaria en temes econòmics de llicencies, i de tenir que instal·lar un entorn tant pesat a totes les màquines on es vulgui executar un programa.

Per solucionar aquest "problema" *National Instruments* ha creat un *framework* anomenat ***RunTime*** que permet executar qualsevol programa implementat en *LabVIEW* sense tenir instal·lat tot l'entorn de programació. Aquest *framework* es pot descarregar de forma gratuïta des de la pagina oficial de *National Instruments* (www.ni.com) i es molt fàcil d'instal·lar i poc pesat en qualsevol màquina amb *SO windows*.

La paraula ***runtime*** té varis significats en termes informàtics, però en aquest cas significa entorn de execució.

Entorn de execució.

Un **entorn de execució** (*runtime enviroment* amb anglès) és un estat de màquina virtual que subministra serveis de software per poder executar certs processos o programes implementats en un llenguatge determinat. En el cas d'aquest treball és el *RunTime* de *LabVIEW*, el qual ens permet executar programes implementats en *LabVIEW* en ordenadors on no estigui instal·lat l'entrorn de programació *LabVIEW*. Pot pertànyer al mateix sistema operatiu, o al software que funciona sota d'ell.

En la majoria dels casos, el sistema operatiu manega la càrrega del programa amb una part del codi que s'anomena carregador, fent configuració bàsica de memòria i enllaçant el programa amb qualsevol biblioteca de vincles dinàmics a la qual faci referència. En alguns casos un llenguatge o implementació farà aquestes tasques en un lloc del *runtime* del llenguatge, a pesar de que és inusual en els llenguatges principals sobre els sistemes operatius de usuaris normals.

4.2. Sistema Gestor de Bases de Dades (SGBD).

Els **Sistemes de gestió de bases de dades** són un tipus de software molt específic, dedicat a servir de interfície entre la base de dades, l'usuari i les aplicacions que la utilitzen. Es compon d'un llenguatge de definició de dades, d'un llenguatge de manipulació de dades i d'un llenguatge de consulta. En els texts o articles que tracten aquest tema, o temes relacionats, es mencionen els termes *SGBD* i/o *DBMS*, sent els dos equivalents, i acrònims, respectivament, de Sistema Gestor de Bases de Dades i *DataBase Management System*, en anglès.

4.2.1. *MS Access 2003.*

Concretament, l'*SGBD* utilitzat en aquest treball és ***MS Access 2003*** que pertany al paquet de programes ***Microsoft Office***, que és una *suite ofimàtica* creada por la empresa *Microsoft*. Funciona oficialment als sistemes operatius *Microsoft Windows* i *Apple Mac OS*, encara que també ho fa en *Linux* si s'utilitza un emulador com *Wine o CrossOver Office*. A més a més les aplicacions inclouen servidors i serveis basats en *Web*.

Microsoft Office és considerat el estàndard "de facto" en programes d'ofimàtica, es per això que es va decidir la utilització d'alguns dels seus programes com és el cas de *MS Access 2003* per la gestió de la *BD*. Com el sistema treballa sota *MS Windows* i un gran percentatge dels usuaris d'aquest *SO* són també usuaris de *MS Office*, s'ha cregut convenient que l'usuari final no hagi d'adquirir programes que no hi siguin al seu abast tant en temes de llicencies com de utilització i funcionament, per aquest motiu es va fer la elecció dels programes ofimàtics al comprovar que complien aquests requisits.

Característiques principals de *MS ACCESS 2003.*

Entre les principals funcionalitats d'A*ccess* es troben:

- Crear taules de dades indexades.
- Modificar taules de dades.
- Relacions entre taules (creació de bases de dades relacionals).
- Creació de consultes i vistes.
- Consultes referències creuades.
- Consultes d'acció (INSERT, DELETE, UPDATE).
- Formularis.
- Informes.
- Crides a la *API* de *windows*.
- Interacció amb altres aplicacions que usen *VBA* (resta de aplicacions de *Microsoft Office, Autocad*, etc.).
- *Macros*.
- Interconexió con entorns de bases de dades de gran nivell (com per exemple *SQL Server*) a través de vinculació.

- Suport de lectura de sistemes de arxius individuals (com *FoxBase* i similars) a través de vinculació i importació de dades.

A més a més, permet crear *frontends* - o programa que mostra la interfície d'usuari - de bases de dades més potents ja que és un sistema capaç d'accedir a taules externes a través d'*ODBC* com si fossin taules *Access*.

4.2.2. Creació d'un *DSN*.

El *Data Source Name* (DSN) o Nom d'Origen de Dades (veure annex A), són uns noms o referències a les bases de dades que utilitzen els sistemes *Windows* per traballar amb aquestes *BDs* per connexió *ODBC*.

En aquest treball es farà servir una BD creada amb *MS Access*, i s'utilitzarà el tipus de connexió *ODBC* per poder-nos comunicar amb ella dins dels programes realitzats.

Per crear el *DSN* pròpiament, hem de anar al Panel de Control de *Windows XP* -> Herramientas Administrativas -> Origenes de datos (ODBC).

4.3. Característiques del Hardware emprat en aquest treball.

En aquest treball s'ha utilitzat un PC amb *SO Windows XP Profesional SP2* i connexió a internet de banda ampla, i amb les següents característiques de *hardware*:

- Processador *Intel Pentium 4 a 2.6 Mhz*.
- Memoria principal: 1GB.
- Tarja *Ethernet* 10/100 *Mbps*.

5 Anàlisi, requeriments i dissenys preliminars

Capítol 5

Anàlisi, requeriments i dissenys preliminars.

5.1. Plantejament del problema.

En aquest treball s'ha plantejat la gestió de les dades generades per un sistema d'automatització d'una granja d'engreix, com s'ha dit anteriorment. El sistema en qüestió constava de varies menjadores, que subministraven el menjar de manera automàtica als animals, les quals tenien una sèrie de sensors, que captaven moltes dades i de molts tipus diàriament. Aquestes dades s'han de captar, gestionar i emmagatzemar per una posterior consulta i/o tractament de les mateixes, i aquí és on comença la tasca principal d'aquest treball.

Fins a aquest moment les dades obtingudes es guardaven en fitxers de text (*.txt*), no hi havia cap base de dades ni cap sistema per gestionar-les.

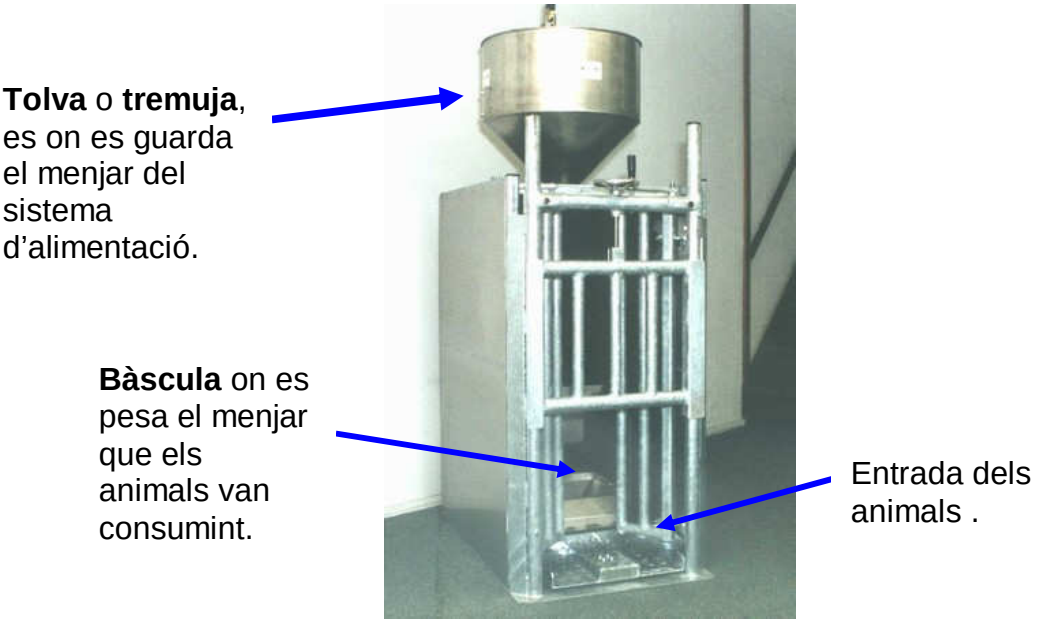
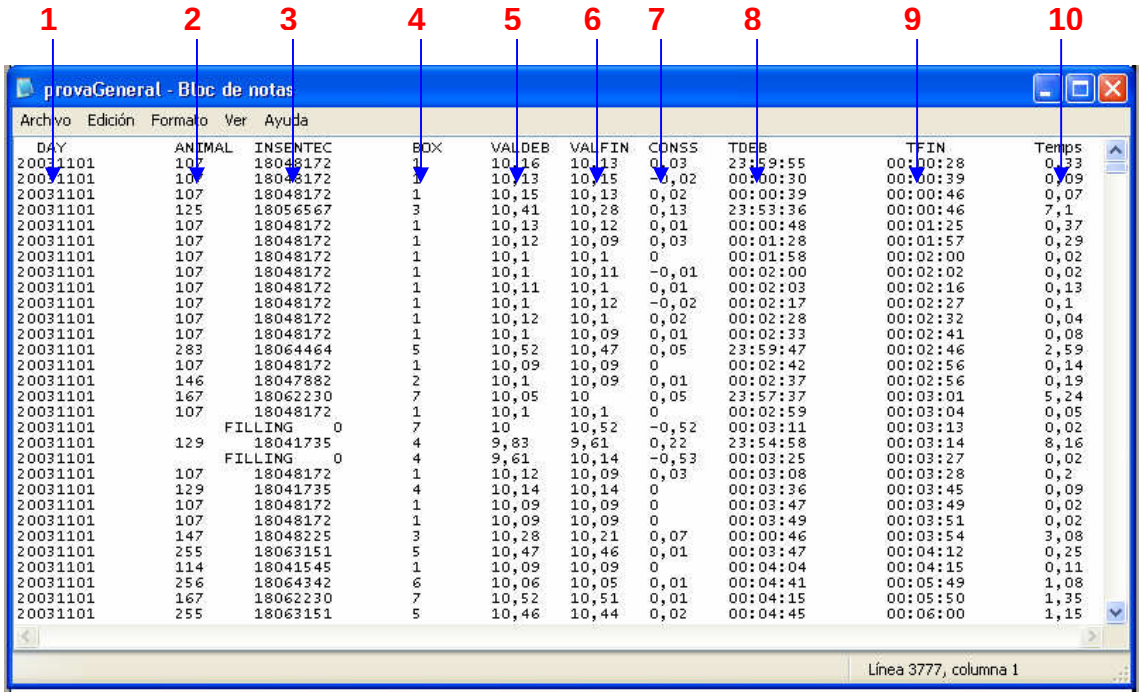


Figura 5.1: Menjadora automàtica del sistema d'alimentació

Es parteix d'una explotació porcina distribuïda en diferents lots (també li podem denominar *box*). Cada lot conté uns 12 animals aproximadament, i els proveeix de menjar amb un sistema d'alimentació automàtic (menjadora, figura 5.1), un per lot. Aquest sistema genera una quantitat important de dades, les quals es guarden en fitxers de text (*.txt*), un per dia *(figura 5.2).* En aquest fitxers s'especifica l'identificador de l'animal, del *box*, l'hora d'entrada i sortida, la data, la quantitat de menjar... Aquests arxius diaris poden arribar a contemplar fins a 5000 registres diaris, les quals no es poden consultar directament, sinó que s'han de mirar els arxius seqüencialment fins que troben les dades que es vulguin inspeccionar.



DAY	ANIMAL	INSENTEC	BOX	VALDEB	VALFIN	CONSS	TDEB	TFIN	Temps
20031101	107	18048172	1	10,16	10,13	0,03	23:59:55	00:00:28	0,33
20031101	107	18048172	1	10,13	10,15	-0,02	00:00:30	00:00:39	0,09
20031101	107	18048172	1	10,15	10,13	0,02	00:00:39	00:00:46	0,07
20031101	125	18056567	3	10,41	10,28	0,13	23:53:36	00:00:46	7,1
20031101	107	18048172	1	10,13	10,12	0,01	00:00:48	00:01:25	0,37
20031101	107	18048172	1	10,12	10,09	0,03	00:01:28	00:01:57	0,29
20031101	107	18048172	1	10,1	10,1	0	00:01:58	00:02:00	0,02
20031101	107	18048172	1	10,1	10,11	-0,01	00:02:00	00:02:02	0,02
20031101	107	18048172	1	10,11	10,1	0,01	00:02:03	00:02:16	0,13
20031101	107	18048172	1	10,1	10,12	-0,02	00:02:17	00:02:27	0,1
20031101	107	18048172	1	10,12	10,1	0,02	00:02:28	00:02:32	0,04
20031101	107	18048172	1	10,1	10,09	0,01	00:02:33	00:02:41	0,08
20031101	283	18064464	5	10,52	10,47	0,05	23:59:47	00:02:46	2,59
20031101	107	18048172	1	10,09	10,09	0	00:02:42	00:02:56	0,14
20031101	146	18047882	2	10,1	10,09	0,01	00:02:37	00:02:56	0,19
20031101	167	18062230	7	10,05	10	0,05	23:57:37	00:03:01	5,24
20031101	107	18048172	1	10,1	10,1	0	00:02:59	00:03:04	0,05
20031101	FILLING	0	7	10	10,52	-0,52	00:03:11	00:03:13	0,02
20031101	129	18041735	4	9,83	9,61	0,22	23:54:58	00:03:14	8,16
20031101	FILLING	0	4	9,61	10,14	-0,53	00:03:25	00:03:27	0,02
20031101	107	18048172	1	10,12	10,09	0,03	00:03:08	00:03:28	0,2
20031101	129	18041735	4	10,14	10,14	0	00:03:36	00:03:45	0,09
20031101	107	18048172	1	10,09	10,09	0	00:03:47	00:03:49	0,02
20031101	107	18048172	1	10,09	10,09	0	00:03:49	00:03:51	0,02
20031101	147	18048225	3	10,28	10,21	0,07	00:00:46	00:03:54	3,08
20031101	255	18063151	5	10,47	10,46	0,01	00:03:47	00:04:12	0,25
20031101	114	18041545	1	10,09	10,09	0	00:04:04	00:04:15	0,11
20031101	256	18064342	6	10,06	10,05	0,01	00:04:41	00:05:49	1,08
20031101	167	18062230	7	10,52	10,51	0,01	00:04:15	00:05:50	1,35
20031101	255	18063151	5	10,46	10,44	0,02	00:04:45	00:06:00	1,15

Figura 5.2: Taula de dades diàries, resultant de la menjadora automàtica.

La taula de dades resultant de l'alimentació dels animals diària, consta de 10 columnes que s'explicaran a continuació:

1. Data (DAY) de l'entrada de l'animal a la menjadora, coincideix amb la data del fitxer.
2. Número identificador de l'animal (ANIMAL).
3. Codi associat al número identificador de l'animal (INSENTEC), és el que identifica la menjadora mitjançant un sensor que detecta el xip de cada animal.

4. *Box* o gàbia on es troben els animals, els quals tenen associada només una menjadora. (Cada *box t*é una menjadora i cada animal només pot pertànyer a un *box*).
5. Pes inicial de la bàscula (VALDEB), on es troba l'aliment dels animals, quan entra un d'ells a menjar. (Hi ha una bàscula per menjadora).
6. Pes final de la bàscula (VALFIN), quan surt l'animal després d'haver ingerit aliment.
7. Pes total (CONSS) d'aliment ingerit per un animal en una visita a la menjadora. (S'obté fent la diferència entre pes final i pes inicial o el que és el mateix VALFIN-VALDEB).
8. Temps inicial (TDEB), hora exacta d'entrada de l'animal a la menjadora, format hh:mm:ss.
9. Temps final (TFIN), hora exacta de sortida de l'animal de la menjadora, format hh:mm:ss.
10. Temps total (TEMPS), diferencia entre TFIN-TDEB, per saber el total de minuts i segons que ha estat l'animal ingerint aliment en cada visita. (Format mm,ss).

Aquests arxius també poden contenir alguns registres erronis o incoherents, com poden ser les visites fantasmes o omplides de menjar al sistema d'alimentació, els quals no es depuren ni es separen de la informació desitjable.

Aquest treball es planteja com una solució a l'**adquisició** i a la **gestió** de les dades generades pel sistema d'alimentació, per facilitar que siguin consultades, analitzades i inspeccionades d'una manera més ràpida i eficient.

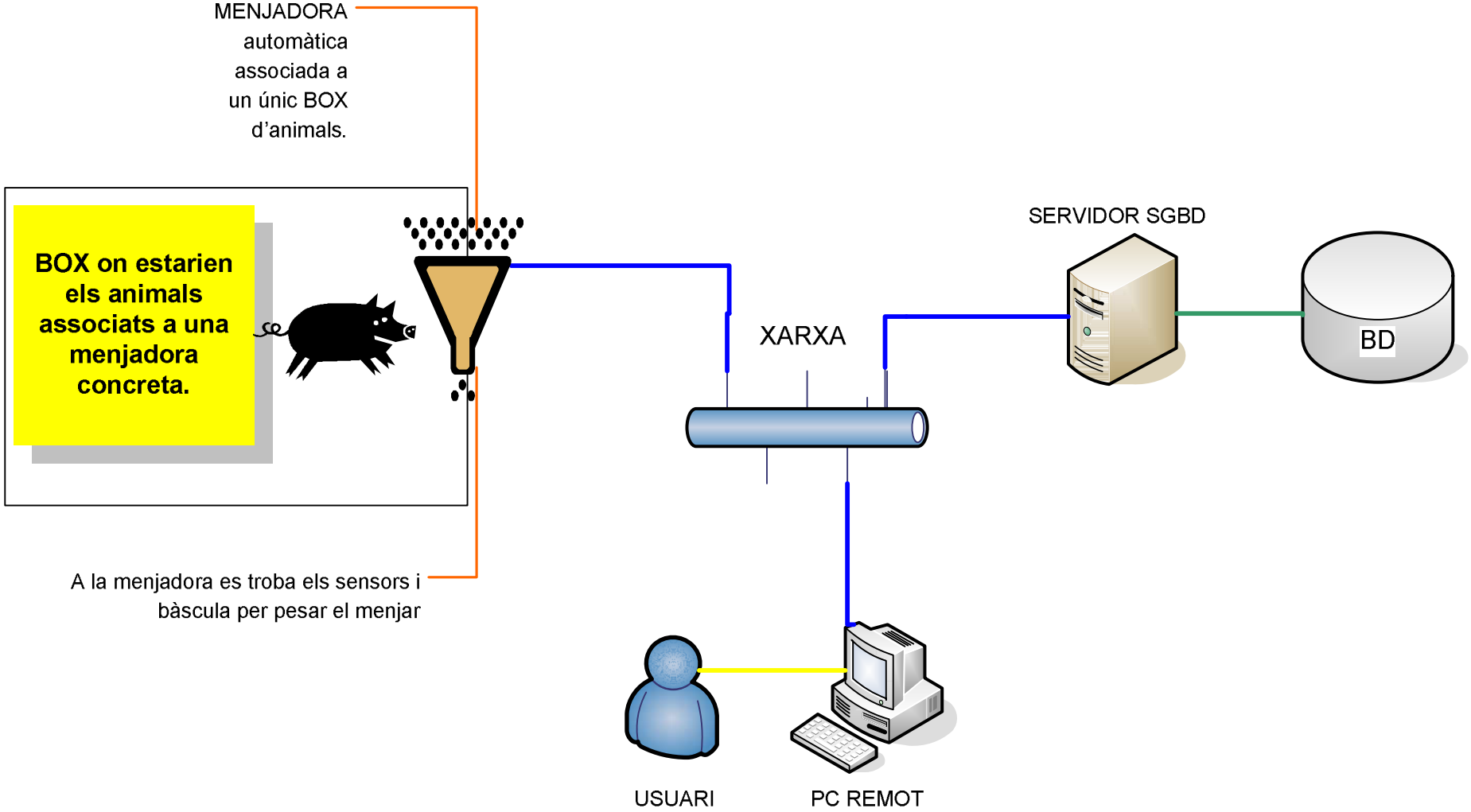


Figura 5.3: Diagrama del disseny preliminar del sistema.

5.1.2. Detecció i depuració de les incidències.

Com s'ha dit anteriorment, la depuració de les dades és un aspecte clau per tal que les consultes i els informes futurs mostrin informació vàlida. Existeixen certs tipus d'incidències, les quals es necessiten detectar i catalogar en un procés previ al desenvolupament del sistema de depuració en qüestió. Sense aquest filtratge no es podrien obtenir informes fiables i reals ja que contindrien registres incoherents que podrien dur a conclusions errònies, és per això que es considera indispensable.

Les incidències que s'han tingut en compte en aquest treball són les següents:

Visites fantasmes *(Ghost Visit)*: Són les visites a la menjadora que no tenen identificador a cap animal, es a dir, que queda registrat una entrada però no se li associa a cap animal de l'explotació. Això pot ser degut a varies causes, com per exemple, l'entrada d'algun petit animal tal com rosegadors, rèptils, insectes... o també pot ser degut a l'entrada d'algun porc petit que hagi passat evitant el sensor. De vegades també pot passar que el sensor de l'animal o "cròtal", que se sol ficar a l'orella a mode d'arracada, es perdi, i degut a aquest motiu fins que es detecta i es reemplaça per un de nou, l'animal continua menjant sense tenir associat cap identificador.

Consums negatius (o nul): Podien ser produïts per animals que entraven a l'estació alimentadora però que no consumien i degut a l'error de $\pm$ 10 g de la bàscula es comptabilitzaven els registres com a negatius. De vegades, els mateixos animals podien orinar o salivar dins la bàscula i fer variar el pes del registre, es a dir, en comptes de disminuir el pes de la bàscula al haver més menjar, podia augmentar.

Consums exagerats: Considerem consum exagerat els que són superiors a 1 Kg. Aquesta incidència normalment és deguda a que l'animal, quan es disposa a menjar, pot ficar una pota a la bàscula, degenerant així el valor registrat.

Registres d'emplenament (*Fillings*): Aquests registres es produïen en el moment en que el sistema reomplia la tremuja i la màquina registrava aquest canvi de pes amb el nom *Filling*. Aquests registres quedaven com a consums negatius, per això s'ha de discriminar entre consums negatius per *Filling* dels que no ho són. Aquesta

acció d'omplir la tremuja es feia cada cop que el pes de menjar disminuïa fins a un valor llindà que podia variar.

Errors d'omplida (Error Filling): Considerem un error d'omplida quan es registra una omplida amb valor nul (pes_total=0), es a dir, quan es realitza un *Filling* que no sigui consum negatiu.

5.2. Disseny de BDs per gestionar tota la informació generada pels animals.

Per gestionar totes les dades generades del sistema es crearan dos bases de dades com em comentat anteriorment. Una d'elles contindrà les dades brutes sense depurar i l'altra contindrà les dades filtrades i llestes per ser inspeccionades i tractades per posteriors informes. Això és degut a que ens interessa tenir les dades en dos llocs, com tenir una copia de seguretat per si es perd o és corrompeix alguna d'elles. Com la memòria de disc és relativament barata, no és cap inconvenient replicar dades del sistema.

5.2.1. Anàlisi conceptual de la base de dades.

Els animals tenen dos identificadors (associats): un tatuat a la pell (*idAnimal*) i l'altre (*insentec*) es troba al *transponder* (chip que porten a enganxat al cos, normalment a l'orella).

El ***idAnimal*** és una variable de tipus numèric, únic i irrepetible, però pot donar-se el cas que al inici de l'experiència (quan s'introdueix l'animal per primera vegada al seu lot corresponent dins l'explotació) es doni automaticament un identificador alfanumèric tal com per exemple: NEW2410NR037.

Posteriorment, a les següents "menjades" de l'animal, s'assigna a un codi numèric associat a un codi *insentec*.

El ***transponder*** és l'**identificador** que llegeix la màquina i s'anomena ***insentec***.

Aquest identificador és únic i irrepetible, i en un principi, està associat a un *idAnimal*, però pot donar-se el cas (un 3% aprox. en cada experiència) que l'animal pugui perdre el seu *transponder* (per diverses causes que no venen al cas) que porta enganxat a l'orella a mode d'arracada.
En aquest cas s'hauria de tornar a fer un nou *transponder*, amb un nou codi *insentec*, ja que la màquina que genera aquests codis no pot generar dos iguals, es a dir, no pot duplicar el codi que ja tenia. Per aquest motiu, el *idAnimal* que ja tenia associat aquell *insentec,* ara tindria un altre *insentec* diferent, però continua sent el mateix animal.
És doncs convenient no fer cap associació física a la BD entre un codi i l'altre.

GhostVisit:
De vegades pot donar-se el cas que el *transponder* de l'animal (codi *insentec*) és perdi o se li mogui de lloc, llavors aquest animal continua estant en el *box* i continua menjant normalment, però no s'enregistren les entrades perquè els sensors no poden llegir el seu codi *insentec*. Això representa una anomalia pel conjunt de les dades, es per això que la màquina enregistra aquesta menjada com *Ghost Visit* o visita fantasma, en la qual queda constància de tota la informació (dia, hora, quantitat de menjar...) però no sabem qui la ha realitzat.
En aquestos casos, de vegades, podem suposar que si un animal no ha menjat en tot un dia, podrien correspondre a aquell animal, llavors s'hauria de comprovar que el *transponder* funcioni correctament i qui ho tingui al lloc adequat per a que els sensors de la màquina ho puguin detectar.

Les causes podrien ser:

- **Pèrdua del *transponder*** per part de l'animal. És possible que els animals és barallin entre ells i el puguin perdre.

- El sensor **no detecta** bé el **codi** de l'animal. Podria ser que tingués algun tipus de brutícia a sobre, que se li hagi mogut del lloc habitual o que entri de costat a la menjadora.

- La màquina conta un animal que ha estat "jugant" amb la palanca de la menjadora, però no ha menjat res.

Això seria sempre una visió aproximada de les vegades que ha entrat l'animal sense identificar.

També es podria donar el cas que cap animal hagués perdut el *transponder* i que hi existeixi alguna "visita" d'altres petits animals, com poden ser rates, i mengin una petita quantitat de menjar, això ho detecta la bàscula i ho conta, però no te cap animal associat.

Box:

El *box* és una gàbia on hi ha una menjadora, en el qual hi ha un numero concret d'animals que sempre seran els mateixos, i estaran identificats i associats a aquell *box*, es a dir, que només podran menjar en aquell *box*, almenys durant la mateixa experiència.

Normalment, aquesta associació es fa durant tota la vida de l'animal, des de que neix fins que va a l'escorxador, es a dir, tot el període d'engreix.

Filling:

La omplida o *filling, és quan es reomple la menjadora.* La tremuja té uns sensors que detecten en tot moment el pes de menjar que hi ha a la bàscula, llavors quan baixa d'un cert pes llindar, automàticament es torna a omplir de menjar fins superar aquest pes llindar.

5.2.2. Disseny de les taules de la base de dades.

A continuació es detallarà el disseny de les BDs amb totes les taules que haurà de tenir cadascuna d'elles.

5.2.2.1. Base de dades bruta.

Es crearà una base de dades amb les dades brutes, tal com surten de la menjadora sense passar cap procés de filtratge. Aquesta *BD* tindrà una estructura composta per una sola taula on es guardaran totes les dades rebudes del sistema d'alimentació. Serà com si passéssim directament els fitxers *.txt* obtinguts dia a dia i els transforméssim en una *BD* sense canviar l'estructura.

La *BD* en qüestió, constarà d'una sola taula amb totes les dades sense depurar:

- dadesbrutes (codi, idAnimal, insentec, box, pes_ini, pes_fi, pes_total, temps_ini, temps_fi, temps_total, data)

  - **codi:** autonumèric (clau primària).
  - **idAnimal:** variable de text (és el codi tatuat de l'animal, sempre és el mateix, però la màquina no ho interpreta, ja que la màquina utilitza el codi *insentec*).
  - **insentec:** variable de text (és el codi digital de l'animal, no sempre és el mateix, ja que si l'animal perd el transponder, se li ha de assignar un altre codi totalment nou i irrepetible; és el que la màquina interpreta).
  - **box:** variable de text (és el codi de la gàbia on es troben els animals).
  - **pes_ini:** variable numèrica *double* (pes inicial de la tremuja on hi ha el menjar, quan l'animal acaba d'entrar i encara no ha menjat).
  - **pes_fi:** variable numèrica *double* (pes final de la tremuja on hi ha el menjar, quan l'animal surt de la menjadora i ja ha menjat).
  - **pes_total:** variable numèrica *double* (pes total menjat per un animal en una entrada a la menjadora, s'obté fent la diferència de [pes_fi – pes_ini]).
  - **temps_ini:** variable de text (és la hora d'entrada de l'animal a la menjadora).
  - **temps_fi:** variable de text (és la hora de sortida de l'animal a la menjadora).
  - **temps_total:** variable de text (és el temps total que l'animal ha estat a la menjadora).
  - **data:** variable de text (la data del dia associada a cada menjada [aaaammdd]).

5.2.2.2. Base de dades depurada.

Es crearà una base de dades amb 5 taules, seran taules independents entre si, i amb una estructura similar, on es guardaran les dades generades pel sistema un cop s'hagin filtrat.

Les taules seran: *DadesEngreix, Filling, GhostVisit, Errors* i *ErrorFilling.*

1. DadesEngreix (codi, idAnimal, insentec, box, pes_ini, pes_fi, pes_total, temps_ini, temps_fi, temps_total, data)
2. Filling (idFilling, box, pes_ini, pes_fi, pes_total, temps_ini, temps_fi, temps_total, data)
3. GhostVisit (idVisit, box, pes_ini, pes_fi, pes_total, temps_ini, temps_fi, temps_total, data)
4. Errors (idError, idAnimal, insentec, box, pes_ini, pes_fi, pes_total, temps_ini, temps_fi, temps_total, data)
5. ErrorFilling (idFilling, box, pes_ini, pes_fi, pes_total, temps_ini, temps_fi, temps_total, data)

La taula 1 (**DadesEngreix**) serà la principal de la *BD*, ja que contindrà tota la informació dels animals pròpiament, ja depurada (sense *filling*, *ghostvisit*, errors ...)

- **codi:** autonumèric (clau primària).
- **idAnimal:** variable de text (és el codi tatuat de l'animal, sempre és el mateix, però la màquina no ho interpreta, ja que la màquina utilitza el codi *insentec*).
- **insentec:** variable de text (és el codi digital de l'animal, no sempre és el mateix, ja que si l'animal perd el transponder, se li ha de assignar un altre codi totalment nou i irrepetible; és el que la màquina interpreta).
- **box:** variable de text (és el codi de la gàbia on es troben els animals).
- **pes_ini:** variable numèrica *double* (pes inicial de la tremuja on hi ha el menjar, quan l'animal acaba d'entrar i encara no ha menjat).
- **pes_fi:** variable numèrica *double* (pes final de la tremuja on hi ha el menjar, quan l'animal surt de la menjadora i ja ha menjat).

- **pes_total:** variable numèrica *double* (pes total menjat per un animal en una entrada a la menjadora, s'obté fent la diferència de [pes_fi – pes_ini]).
- **temps_ini:** variable de text (és la hora d'entrada de l'animal a la menjadora).
- **temps_fi:** variable de text (és la hora de sortida de l'animal a la menjadora).
- **temps_total:** variable de text (és el temps total que l'animal ha estat a la menjadora).
- **data:** variable de text (la data del dia associada a cada menjada [aaaammdd]).

La taula 2 (**Filling**) emmagatzemarà les dades de les omplides o *fillings* a la *BD*.

- **idFilling:** autonumèric (clau primària).
- **box:** variable de text (és el codi de la gàbia on es troba la menjadora que s'ompliprà).
- **pes_ini:** variable numèrica *double* (pes inicial de la tremuja on hi ha el menjar, quan la màquina començar a actuar i omplir-la).
- **pes_fi:** variable numèrica *double* (pes final de la tremuja on hi ha el menjar, quan la màquina ja ha acabat d'omplir-la).
- **pes_total:** variable numèrica *double* (pes total que s'ha omplert la tremuja en un *filling* a la menjadora, s'obté fent la diferència de [pes_fi – pes_ini]).
- **temps_ini:** variable de text (és la hora que comença el *filling* a la menjadora).
- **temps_fi:** variable de text (és la hora que acaba el *filling* a la menjadora).
- **temps_total:** variable de text (és el temps total en que ha actuat el *filling* en aquesta menjadora).
- **data:** variable de text (la data del dia associada a cada *filling* [aaaammdd]).

La taula 3 (**GhostVisit**) emmagatzemarà les dades de les visites fantasmes o *ghost visit* a la *BD*.

- **idVisit:** autonumèric (clau primària).
- **box:** variable de text (és el codi de la gàbia on s'ha produït la visita fantasma).

- **pes_ini:** variable numèrica *double* (pes inicial de la tremuja on hi ha el menjar, quan comença la visita fantasma).
- **pes_fi:** variable numèrica *double* (pes final de la tremuja on hi ha el menjar, quan acaba la visita fantasma).
- **pes_total:** variable numèrica *double* (pes total que s'ha consumit de la tremuja en una visita fantasma, s'obté fent la diferència de [pes_fi – pes_ini]).
- **temps_ini:** variable de text (és la hora que comença la visita fantasma a la menjadora).
- **temps_fi:** variable de text (és la hora que acaba la visita fantasma a la menjadora).
- **temps_total:** variable de text (és el temps total de la visita fantasma en aquesta menjadora).
- **data:** variable de text (la data del dia associada a cada *visita fantasma* [aaaammdd]).

La taula 4 (**Errors**) contindrà la informació dels errors (consum negatiu, consum exagerat...) de les dades del sistema d'alimentació.

- **idError:** autonumèric (clau primària).
- ***idAnimal*:** (clau forana) variable de text (és el codi tatuat de l'animal, sempre és el mateix, però la màquina no ho interpreta, ja que la màquina utilitza el codi *insentec*).
- **insentec:** variable de text (és el codi digital de l'animal, no sempre és el mateix, ja que si l'animal perd el transponder, se li ha de assignar un altre codi totalment nou i irrepetible; és el que la màquina interpreta).
- **box:** variable de text (és el codi de la gàbia on es troben els animals).
- **pes_ini:** variable numèrica *double* (pes inicial de la tremuja on hi ha el menjar, quan l'animal acaba d'entrar i encara no ha menjat).
- **pes_fi:** variable numèrica *double* (pes final de la tremuja on hi ha el menjar, quan l'animal surt de la menjadora i ja ha menjat).
- **pes_total:** variable numèrica *double* (pes total menjat per un animal en una entrada a la menjadora, s'obté fent la diferència de [pes_fi – pes_ini]).

- **temps_ini:** variable de text (és la hora d'entrada de l'animal a la menjadora).
- **temps_fi:** variable de text (és la hora de sortida de l'animal a la menjadora).
- **temps_total:** variable de text (és el temps total que l'animal ha estat a la menjadora).
- **data:** variable de text (la data del dia associada a cada menjada [aaaammdd]).

La taula 5 (**ErrorFilling**) emmagatzemarà les dades errònies de les omplides o *Fillings* a la *BD*.

- **idFilling:** autonumèric (clau primària).
- **box:** variable de text (és el codi de la gàbia on s'ha produït la visita fantasma).
- **pes_ini:** variable numèrica *double* (pes inicial de la tremuja on hi ha el menjar, quan comença la visita fantasma).
- **pes_fi:** variable numèrica *double* (pes final de la tremuja on hi ha el menjar, quan acaba la visita fantasma).
- **pes_total:** variable numèrica *double* (pes total que s'ha consumit de la tremuja en una visita fantasma, s'obté fent la diferència de [pes_fi – pes_ini]).
- **temps_ini:** variable de text (és la hora que comença la visita fantasma a la menjadora).
- **temps_fi:** variable de text (és la hora que acaba la visita fantasma a la menjadora).
- **temps_total:** variable de text (és el temps total de la visita fantasma en aquesta menjadora).
- **data:** variable de text (la data del dia associada a cada *visita fantasma* [aaaammdd]).

5.3. Automatització de la informació generada pels animals.

En aquest apartat s'explicarà els procediments que s'han realitzat en aquest treball per dur a terme els objectius exposats anteriorment.

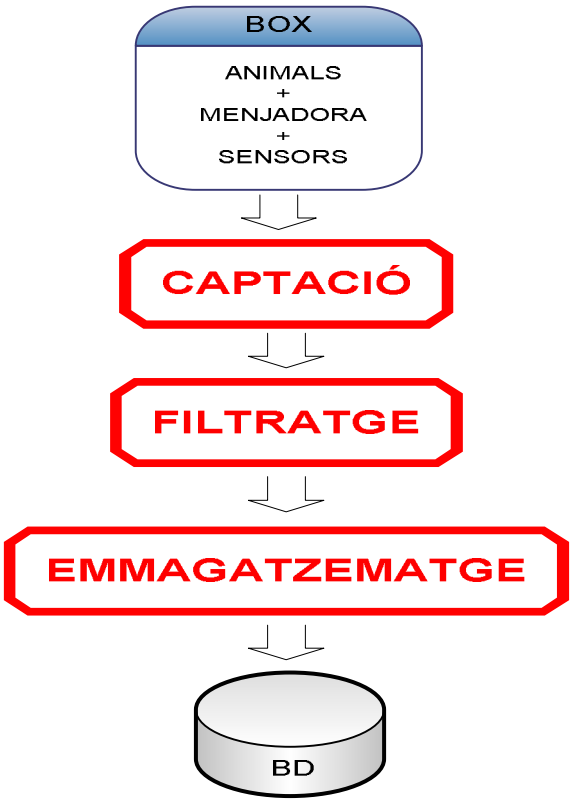


Figura 5.4: Esquema dels procediments que durà a terme el sistema.

5.3.1. Captació i depuració de les dades.

Aquest és el primer punt clau del treball, es tracta de realitzar un programa el qual capti les dades des d'un fitxer de text (*.txt*) que conté totes les dades generades pel sistema d'alimentació en un mateix dia. Posteriorment es separaran per poder inserir-les a les bases de dades, tant a la bruta, com al la depurada un cop s'han filtrat. Tant el procés de captació com de filtrat es realitzaran en un mateix programa implementat en *LabVIEW*, per requeriments del treball.

5.3.2. Emmagatzemament en la base de dades.

Com hem dit anteriorment gestionarem les dades obtingudes pel sistema d'alimentació en dues bases de dades. Aquestes bases de dades funcionaran de manera independent, i estaran creades i gestionades pel mateix *Sistema Gestor de BDs.*

5.3.2.1. Des de fitxer de text *(.txt)*.

Aquest procés serà la continuació directa de l'anterior programa de captació i depuració de les dades, sumant-li l'emmagatzemament a les bases de dades. Aprofitant el programa el programa anterior, ho ampliarem per tal de afegir la funcionalitat d'emmagatzemar les dades captades i filtrades a la base de dades. De fet seran dos programes, molt similars, un que guardi les dades a la *BD* bruta, i l'altre que ho faci a la *BD* depurada.

Partirem de la base dels fitxers *.txt* obtinguts de la menjadora automàtica, per poder anar inserint línia a línia a les taules corresponents de la *BD* en cada cas.

5.3.2.2. Simulació d'emmagatzemament a temps real amb *DataSocket*.

Tenint en compte que al sistema global tots els instruments (programes) estaran comunicats amb *DataSocket*, per requeriments del treball, es va adaptar el programa d'emmagatzemament de les dades a dos nous programes connectats amb *DSTP* de *LabVIEW*. Aquests programes tracten de simular la situació real que es portarà a terme quan estigui implantat amb el sistema general.

Constarà de dos programes amb funcions molt específiques:

1) SEND (enviament): serà l'encarregat de anar llegint les dades del fitxers *.txt* per poder enviar línia a línia tot el contingut del mateix.

2) RECIEVE (rebuda): aquest programa s'encarregarà de rebre les dades, quan l'altre programa li enviï línia a línia, simulant així l'enviament i rebuda de dades "al vol" i guardant-les a la *BD*.

5.3.3. Consulta a la *BD*.

Ens trobem en el segon punt clau del treball, en el qual em d'aconseguir un programa independent als altres, el qual sigui capaç de connectar-se a les nostres bases de dades per poder realitzar consultes.

És en aquest moment quan em de fer servir, a banda de les funcions normals de *LabVIEW 7.1*, les funcions específiques del *toolkit* de bases de dades. Es vol implementar un programa el qual pugui realitzar tres tipus de consultes:

1) Consultes predefinides: en el qual l'usuari només tingui que escriure la data de la consulta que vol realitzar.

2) Consultes guiades: en aquest tipus de consultes l'usuari anirà configurant la seva consulta, ja en llenguatge SQL, mitjançant despleglables amb els quals ha de triar cada opció del: SELECT, FROM, WHERE.

3) Consultes avançades: aquestes seran les consultes per usuaris avançats i coneixedors del llenguatge SQL, ja que es tracta d'escriure la consulta completament en aquest llenguatge.

5.3.3.1. Consulta telemàtica amb un servidor web.

Aquest programa serà una ampliació de l'anterior, també farà els mateixos tipus de consultes amb l'afegit que es podran fer remotament, utilitzant algun servidor web compatible amb els programes implementats en *LabVIEW*. Per fer això és necessitarà una màquina que faci de servidor, on estarà funcionant un programa servidor de webs, un cop fet això, es podrà anar a una màquina remota connectada a

la xarxa, i mitjançant un navegador, ficar l'adreça IP del PC servidor i el nom del *.vi* (virtual instrument). Aquest instrument virtual es pot controlar remotament des del navegador com si s'estigués treballant localment.

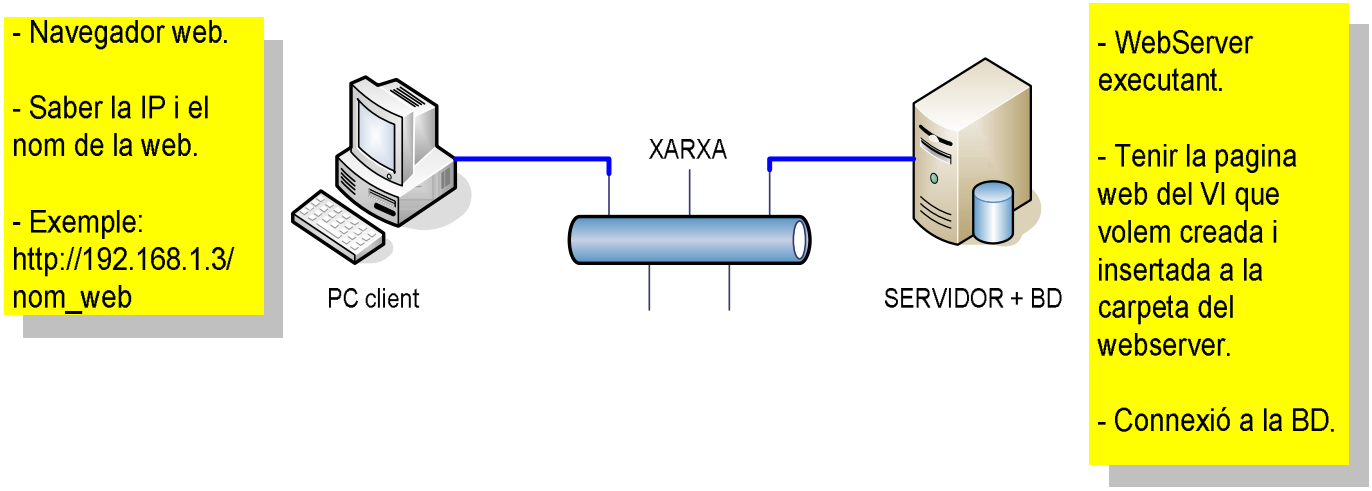


Figura 5.5: Diagrama de la consulta telemàtica a la BD mitjançant un servidor web.

5.3.4. Generació d'informes.

Aquest és l'últim objectiu principal del treball, i es tracta de poder realitzar informes partint de les dades que em registrat anteriorment a la base de dades. Per fer-ho s'utilitzarà un programa de generació d'informes que sigui en tot moment compatible amb la resta de programes del sistema global implementat en *LabVIEW*.

Aquests informes es faran mitjançant una consulta a la base de dades, per després poder triar les característiques i crear-los com es mostra a la *figura 5.5*. Un cop triats tots els paràmetres que es volen, es generarà un informe, el qual es podrà guardar localment, imprimir, convertir a *.pdf*...

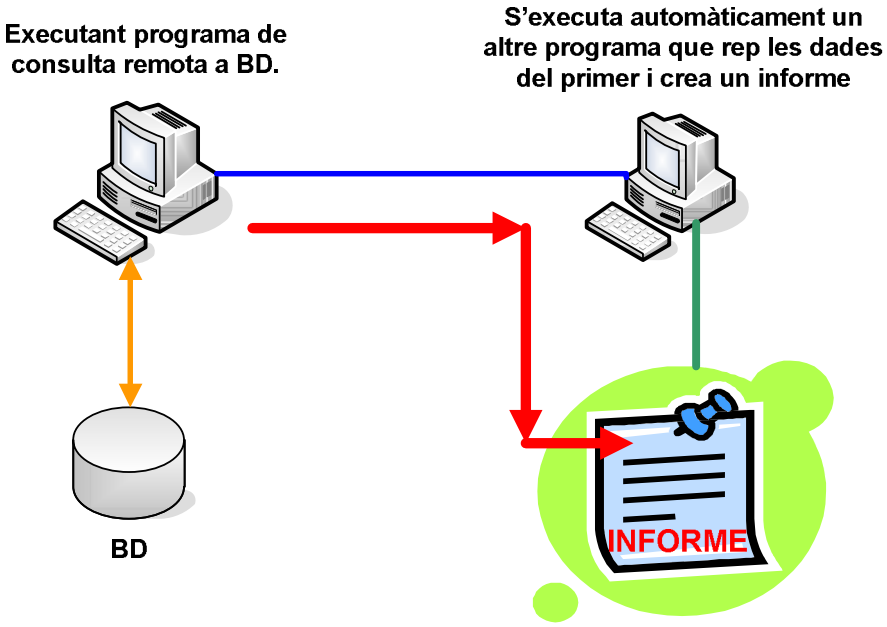


Figura 5.6: Diagrama del disseny preliminar de la creació d'informes.

5.3.4.1. Generació d'informes via *Internet*, lligat a una consulta.

Un pas més al programa anterior és poder realitzar aquest informes des d'una màquina remota. Aquesta eina s'implementarà amb l'ajuda d'algun servidor web compatible amb els programes del sistema implementats en *LabVIEW*, el qual ens permeti les mateixes funcionalitats que en el programa de generació d'informes localment (*figura 5.6*).

6 Disseny i implementació

Capítol 6

Disseny i implementació.

En aquesta part del treball s'explicarà tant el disseny de la interficie i la implementació dels programes en qüestió, com els diagrames amb el disseny de la connexió entre si, i amb el sistema global (*figura 6-A*).

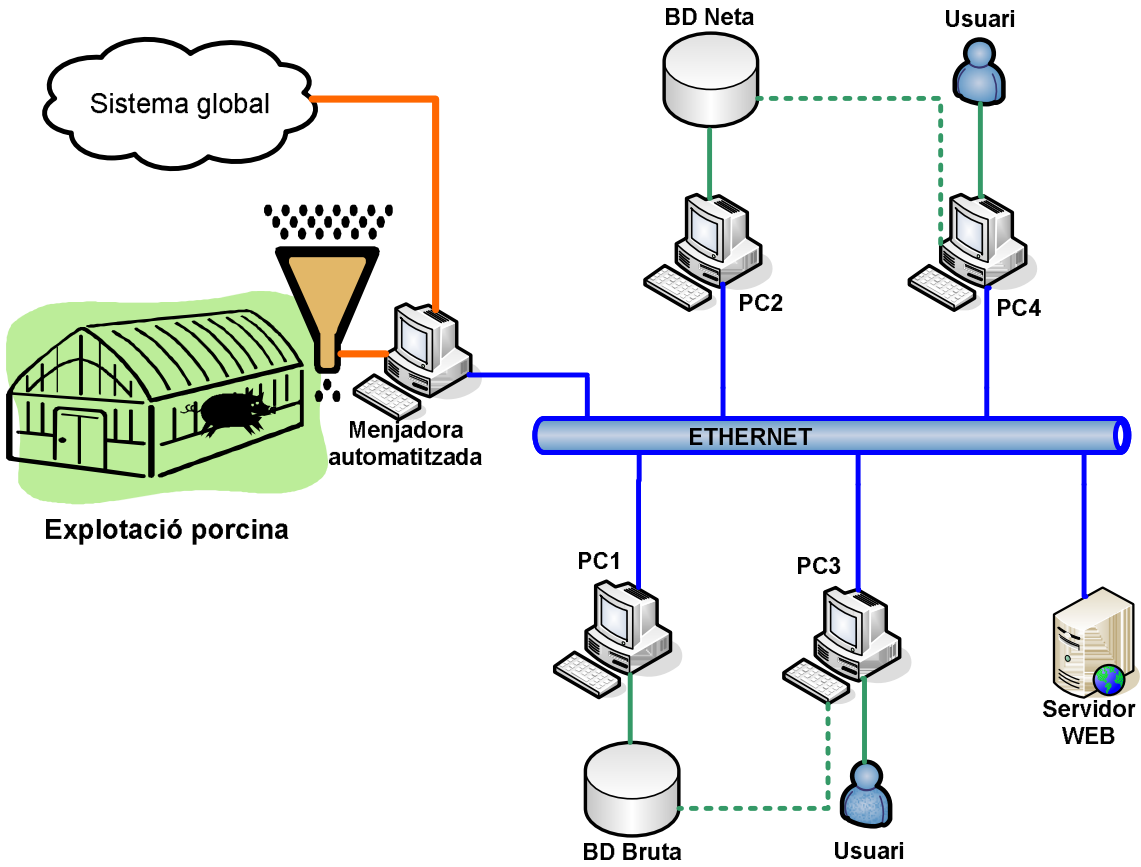


Figura 6-A: Diagrama de connexió de les diferents eines implementades en el treball.

Al *PC1* s'executarà el programa de inserció a la *BD bruta* i la BD propiament; al *PC2* s'executarà el programa de filtració i d'inserció a la *BD neta* i la BD amb aquestes dades depurades; al *PC3* s'executarà el programa de consultes a la BD; al *PC4* s'executarà el programa de generació d'informes i al *servidor web* hi haurà els *VIs* en format web dels programes de consulta a la BD i de generació d'informes.

Aquesta és una possible configuració de les eines implementades, però en poden haber bastantes més, com per exemple, si fós una explotació petita, podria executar tots els programes en un sol PC.

6.1. Eina de depuració de dades brutes i inserció a la *BD*.

L'eina de depuració de dades brutes i inserció a la base de dades (*figura 6-B*), es desglossa en dos programes que a la vegada contenen varis sub-programes. La funcionalitat principal de la primera eina és la inserció de les dades sense depurar a una base de dades bruta o no es discrimina cap dada que arribi del sistema d'alimentació. La segona eina, depuració de dades amb inserció inclosa, es basa inserir les dades a una BD, passant per un filtre previ on es separen les dades d'engreix principals de les dades errònies, omplides de menjar, visites fantasmes...

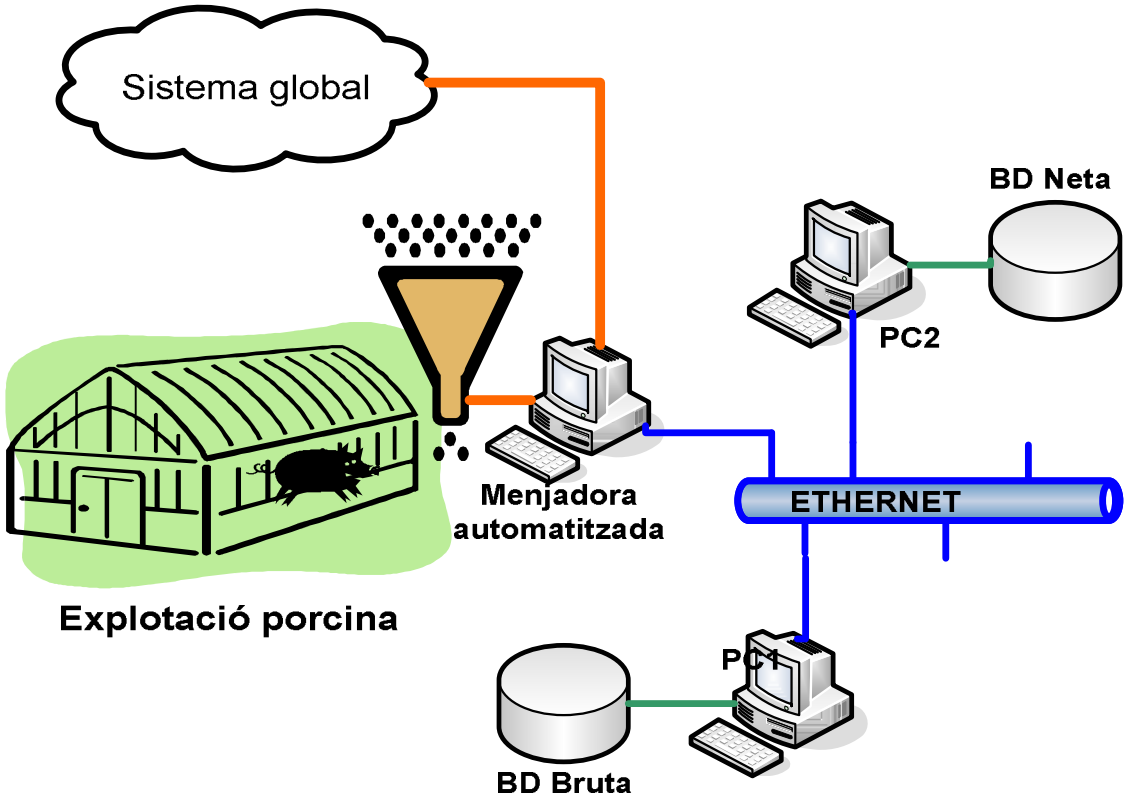


Figura 6-B: Diagrama de les eines de inserció i de depuració amb les respectives BDs.

6.1.1. Inserció a la BD sense depurar les dades brutes.

insertBDdef.vi (programa principal)

Aquest programa serveix per inserir totes les dades del fitxer *.txt* seleccionat a la la Base de Dades (BDbruta) que hem creat anteriorment amb una sola taula anomenada "dadesbrutes" i amb els següents camps:

1. codi (autonumeric, clau primària)
2. idAnimal (char)
3. insentec (char)
4. box (char)
5. pes_ini (double)
6. pes_fi (double)
7. pes_total (double)
8. hora_ent (char)
9. hora_sort (char)
10. temps_total (char)
11. data (char)

Al executar el programa, aquest ens demanarà que triem primer la BD on volem inserir les dades, juntament amb el seu *DSN*. Posteriorment triem el fitxer que servirà de font de dades a inserir.

Front Panel

Path on triarà el fitxer que volem llegir.

Controlador de la velocitat d'execució.

Text box on s'anirà visualitzant les linies llegides del fitxer *.txt.*

Línia de camps llegits del fitxer de text actual.

Indicador de possibles errors de la BD.

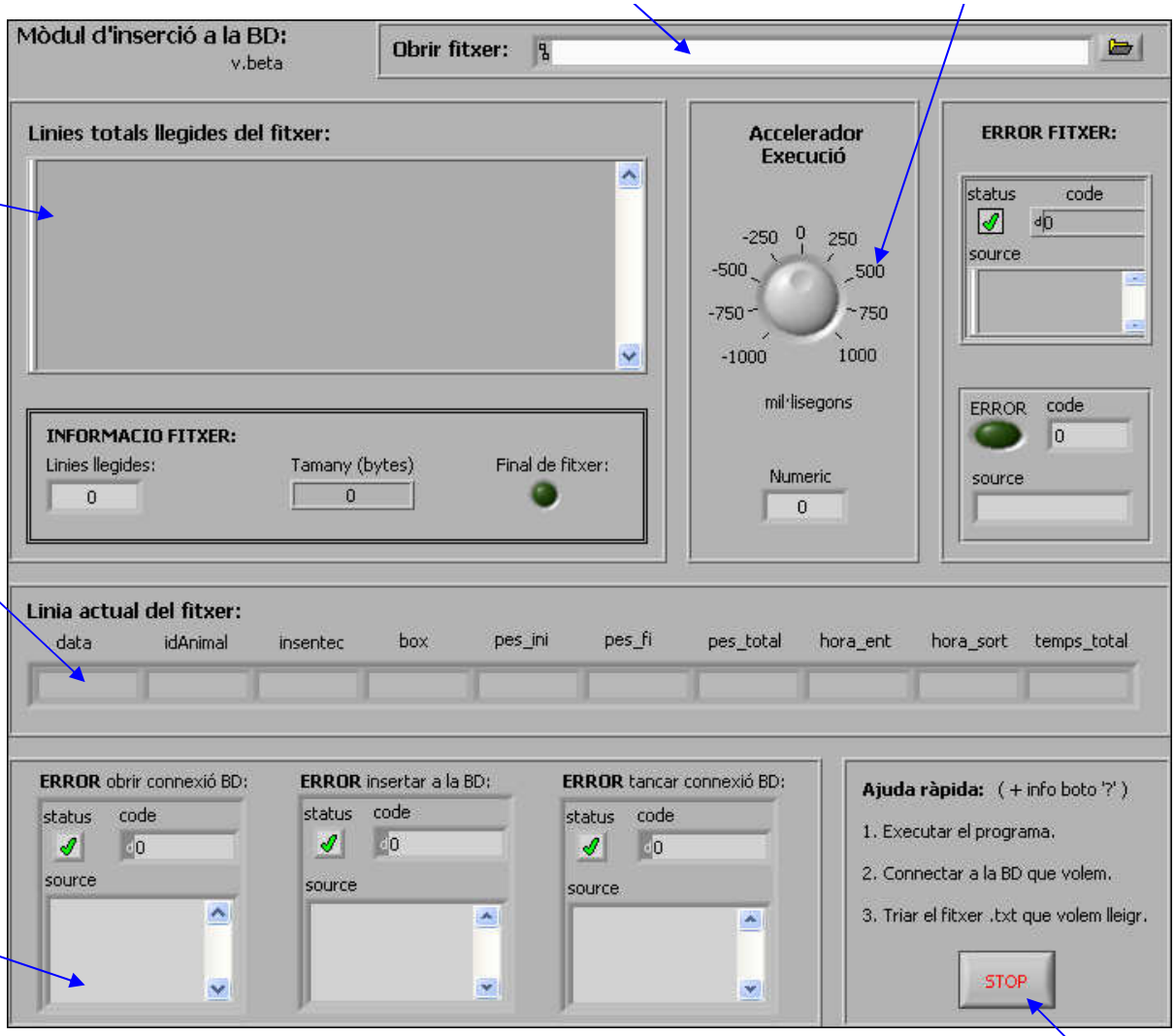
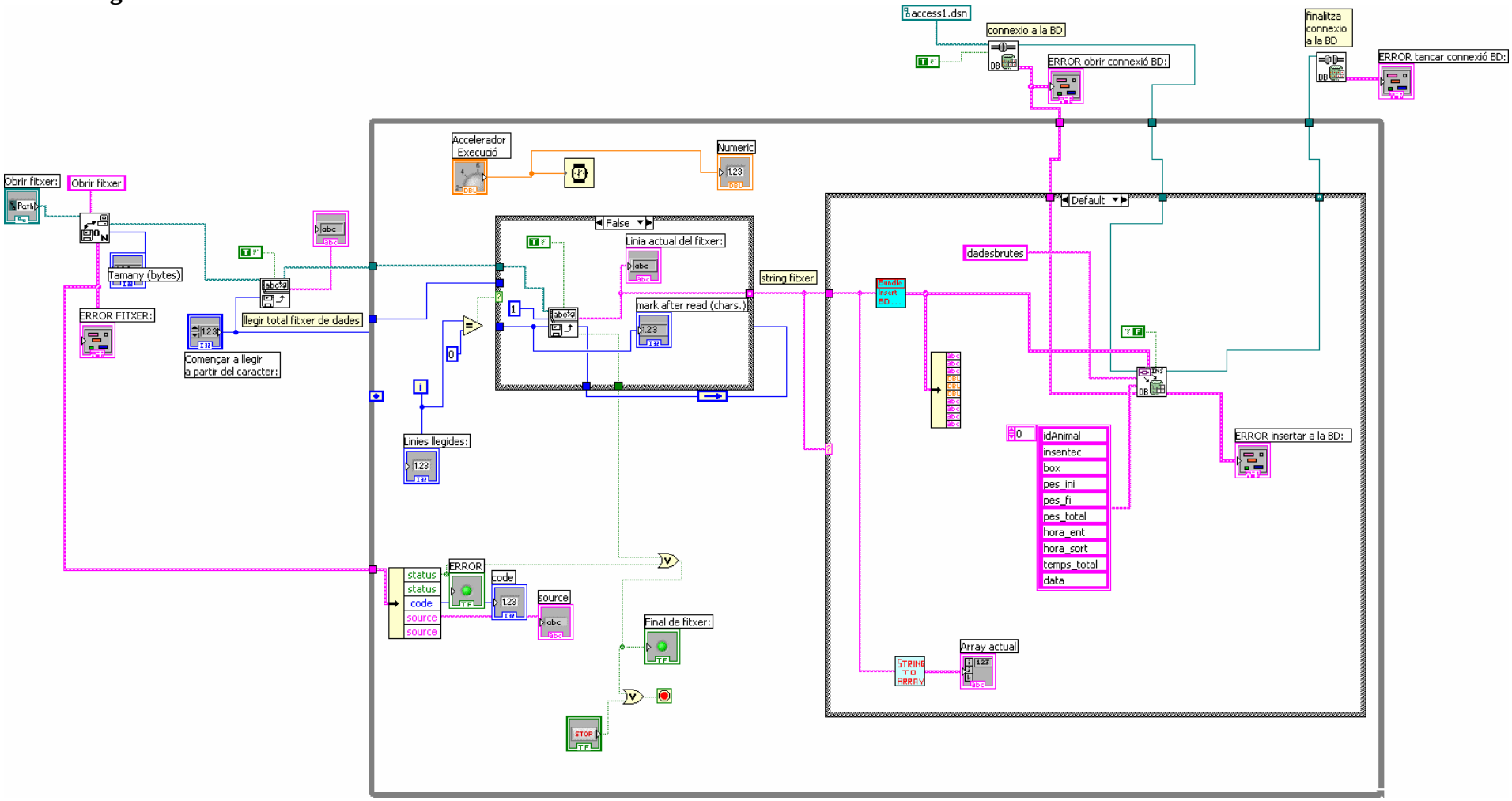


Figura 6.1-a: Panel frontal (interfície) del programa *insertBDdef.vi.*

Botó per parar la execució del programa.

Block Diagram



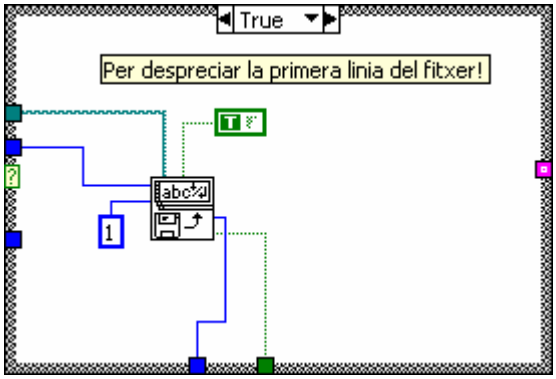


Figura 6.1-b: Block Diagram (codi) del programa *insertBDdef.vi.*

Llistat de subVIs del programa insertBDdef.vi

Read Lines From File.vi
Llegeix un fitxer de text línia per línia.

Open/Create/Replace File.vi
Obrir, crear o reemplaçar un fitxer.

DB Tools Open Connection.vi
Obrir la connexió a la BD.

DB Tools Insert Data.vi
Insertar dades a una BD.

DB Tools Close Connection.vi
Tancar la connexió a la BD.

subVI-insertBDdef.vi
Separar els camps de la línia de text i ficar-los en una matriu per inserir-los posteriorment a la BD.

subVI-stringToArray.vi
Transformar un *string* (cadena de caracters) en un *array* o matriu.

subVI-insertBDdef.vi

Aquest subprograma el que fa es separar cada *string* de la línia actual que arriba del fitxer, en els camps corresponents a la BD per poder inserir-los posteriorment a cada columna de la taula corresponent.

Connector Pane

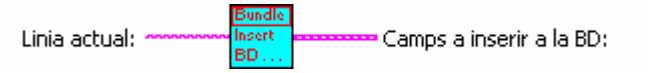


Figura 6.2-a: Connector Pane del *subVI-insertBDdef.vi*.

Front Panel

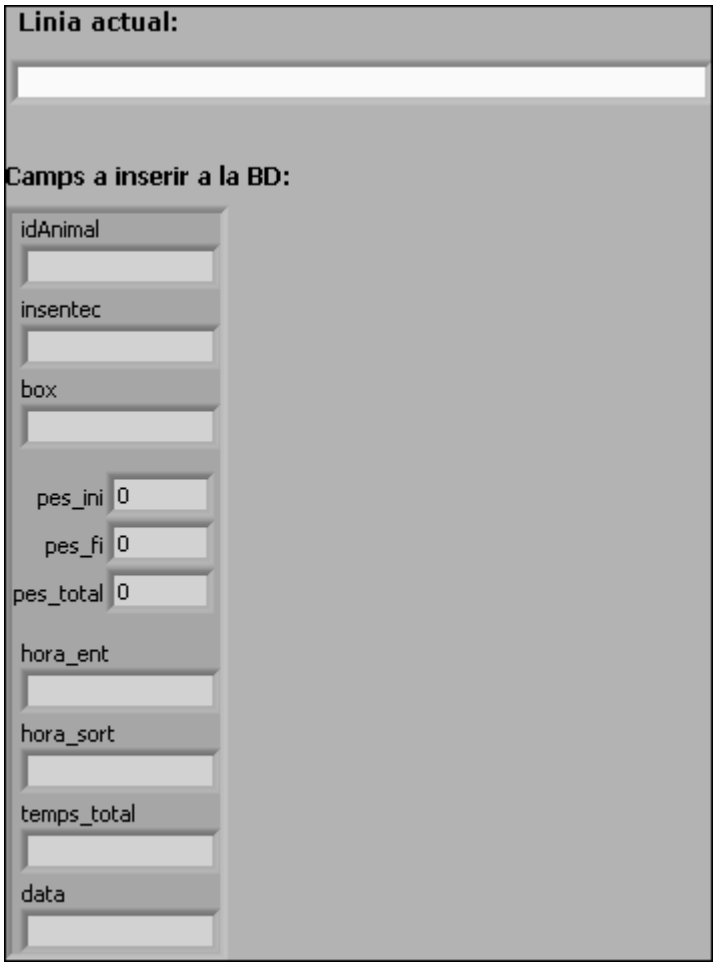


Figura 6.2-b: Front Panel del *subVI-insertBDdef.vi*.

Block Diagram

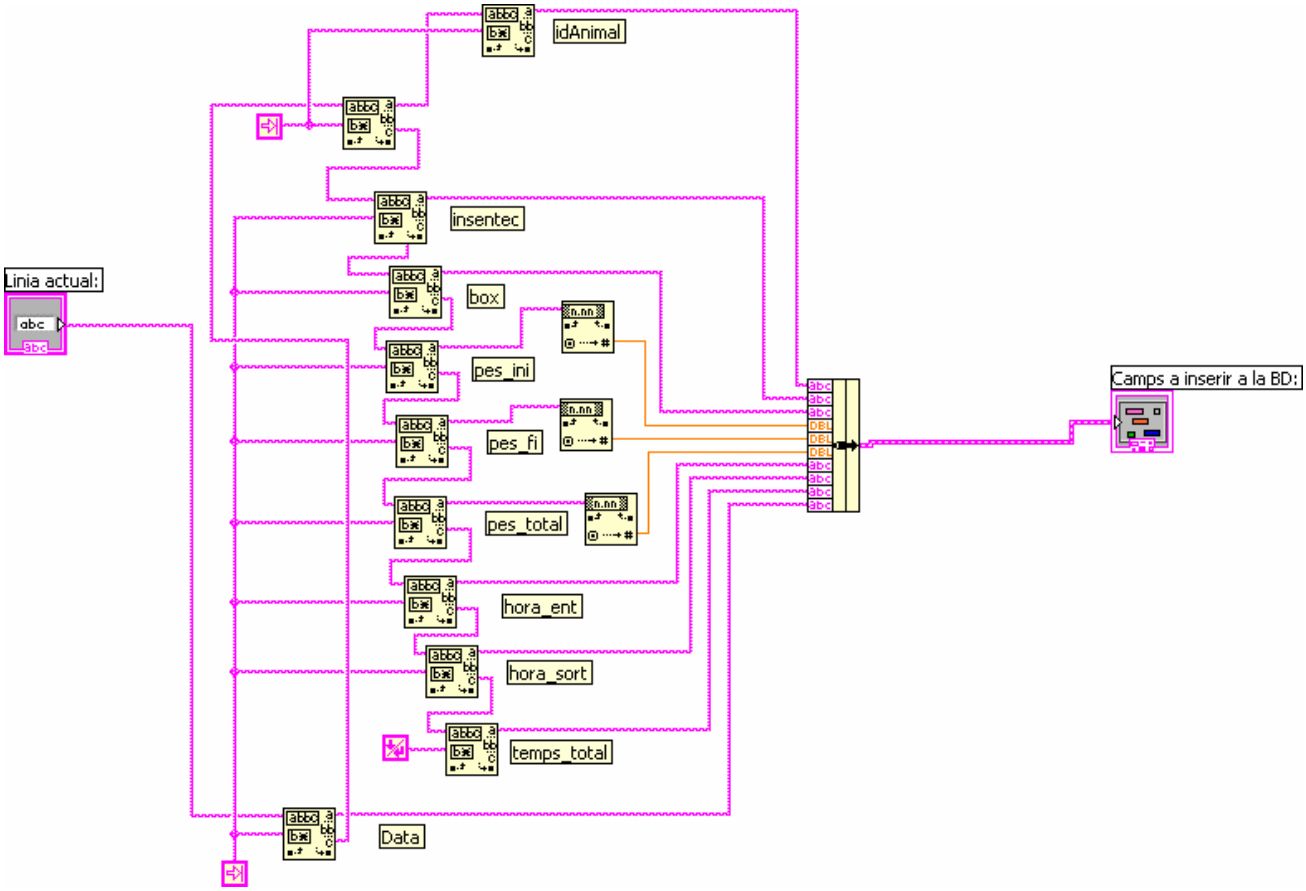


Figura 6.2-c: Block Diagram del *subVI-insertBDdef.vi*.

subVI-stringToArray.vi_____

Aquest subprograma el que fa es transformar cada *string* (caràcters) de la línia actual que arriba del fitxer, en un *Array* (matriu) per millorar la posterior inserció a la BD.

Connector Pane

Figura 6.3: Connector Pane (icona amb els connectors) del *subVI-stringToArray*.

Front Panel

Figura 6.4-a: Front Panel del *subVI-stringToArray*.

Block Diagram

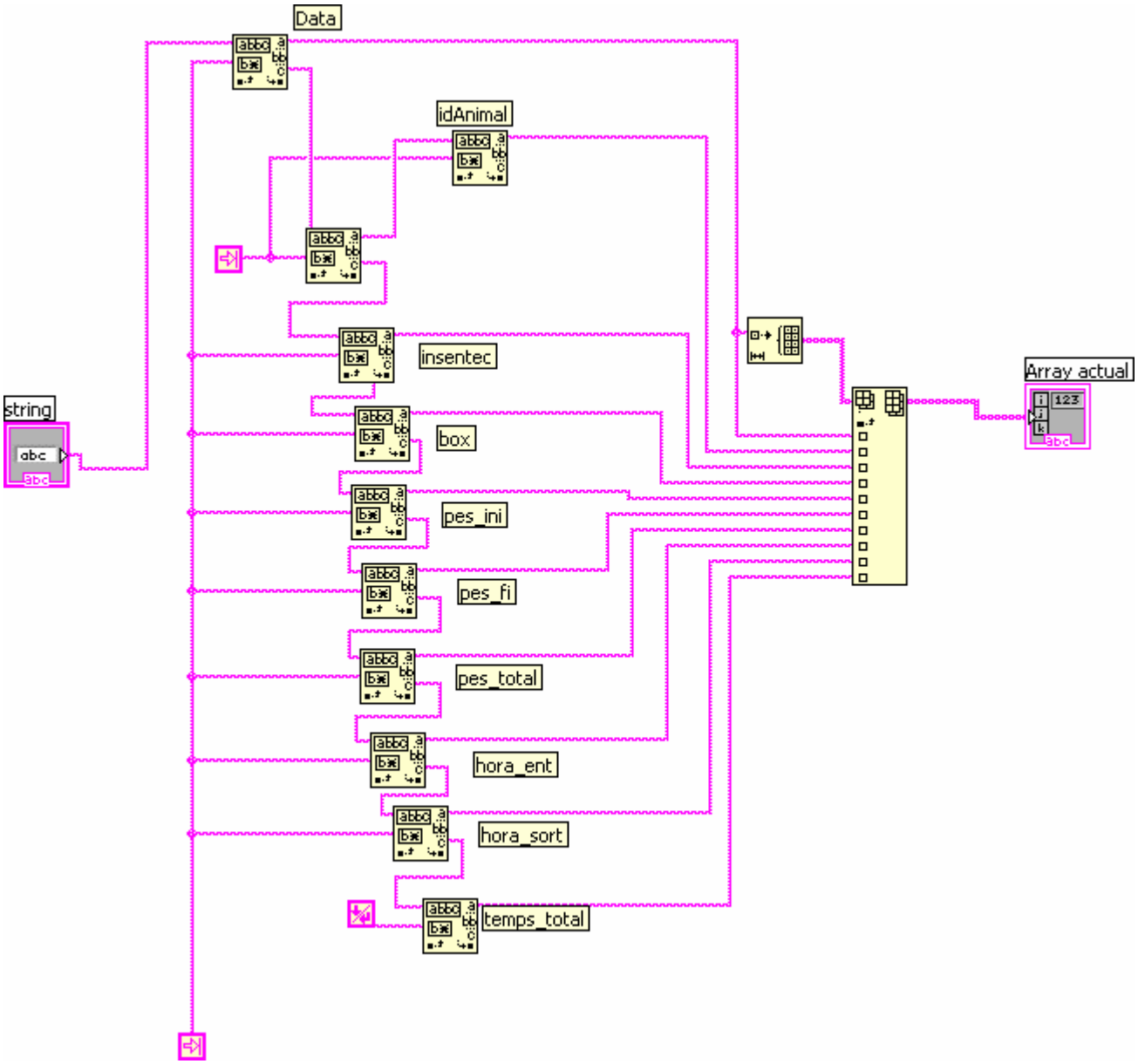


Figura 6.4-b: Block Diagram del *subVI-stringToArray*.

6.1.2. Filtració de les dades per la posterior inserció a la BD depurada.

depuraBDdef.vi (programa principal)
Aquest programa el que pretén és separar les dades que entren d'un fitxer de dades barrejades i amb possibles errors, per insertar-les posteriorment a una BD depurada dividida en cinc taules:

1. DadesEngreix
   (conjunt de dades rellevants dels animals)

2. Filling
   (dades que es recullen despres de cada omplida de les menjadores)

3. GhostVisit
   (per enregistrar les visites fantasmes)

4. Errors
   (dades erronies que recull el sistema com són:
   - Consum negatiu: pes_total<0
   - Consum nul: pes_total=0
   - Consum exagerat: pes_total>1)

5. ErrorFilling
   (dades errornies de les linies de Filling
   -Omplida nul·la: pes_total>1)

Al executar el programa, aquest ens demanarà que triem primer la BD on volem inserir les dades depurades, i després el fitxer que servirà de font de dades a depurar i a inserir.

Front Panel

Com es pot observar a continuació, aquest programa és molt semblant a l'anterior de inserció de dades a la BD, però en aquest es discrimina les dades que s'inserten i on s'inserten a la base de dades.

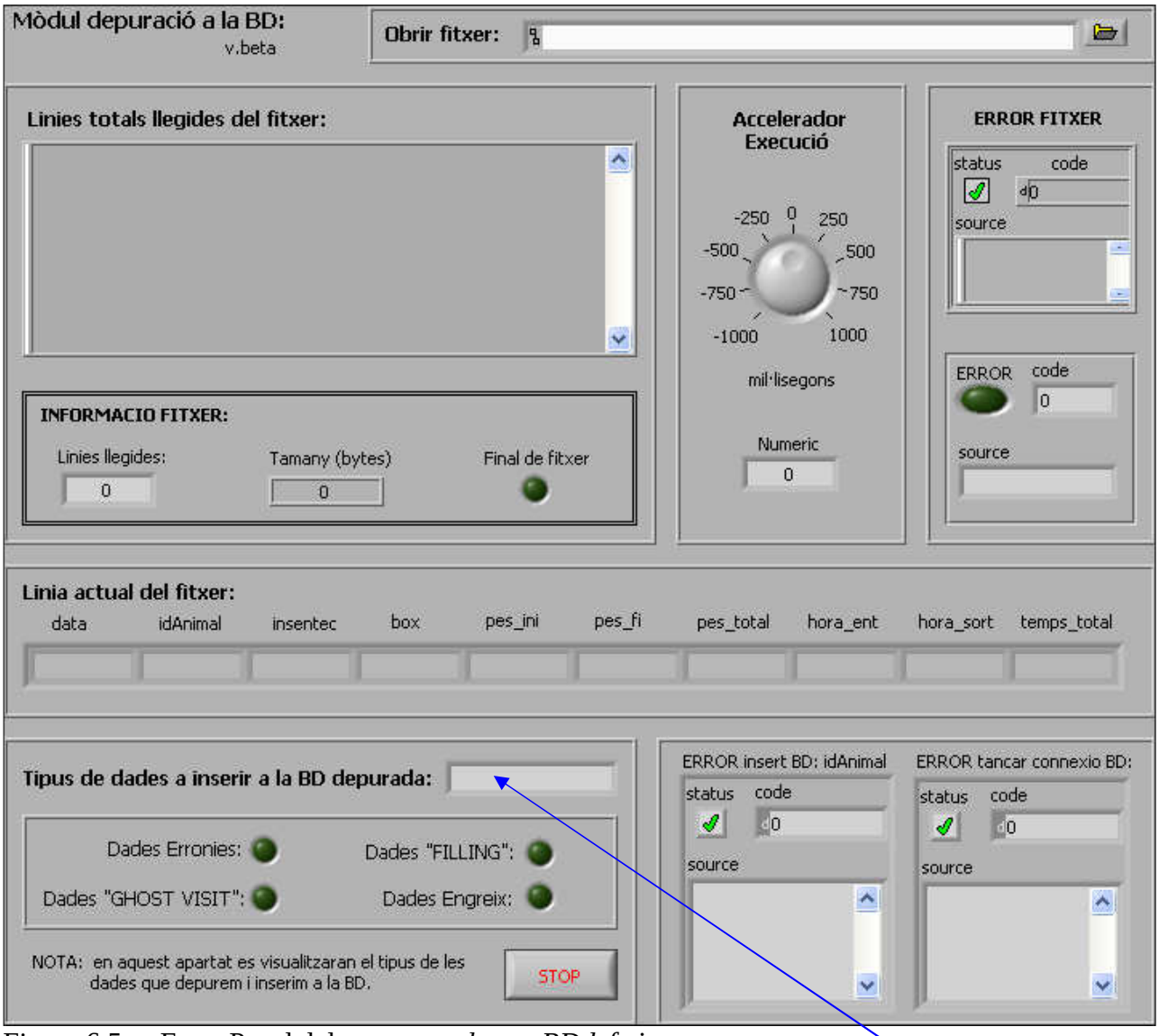


Figura 6.5-a: Front Panel del programa *depuraBDdef.vi.*

Tipus de dades que es vol insertar a la BD depurada.

Block Diagram

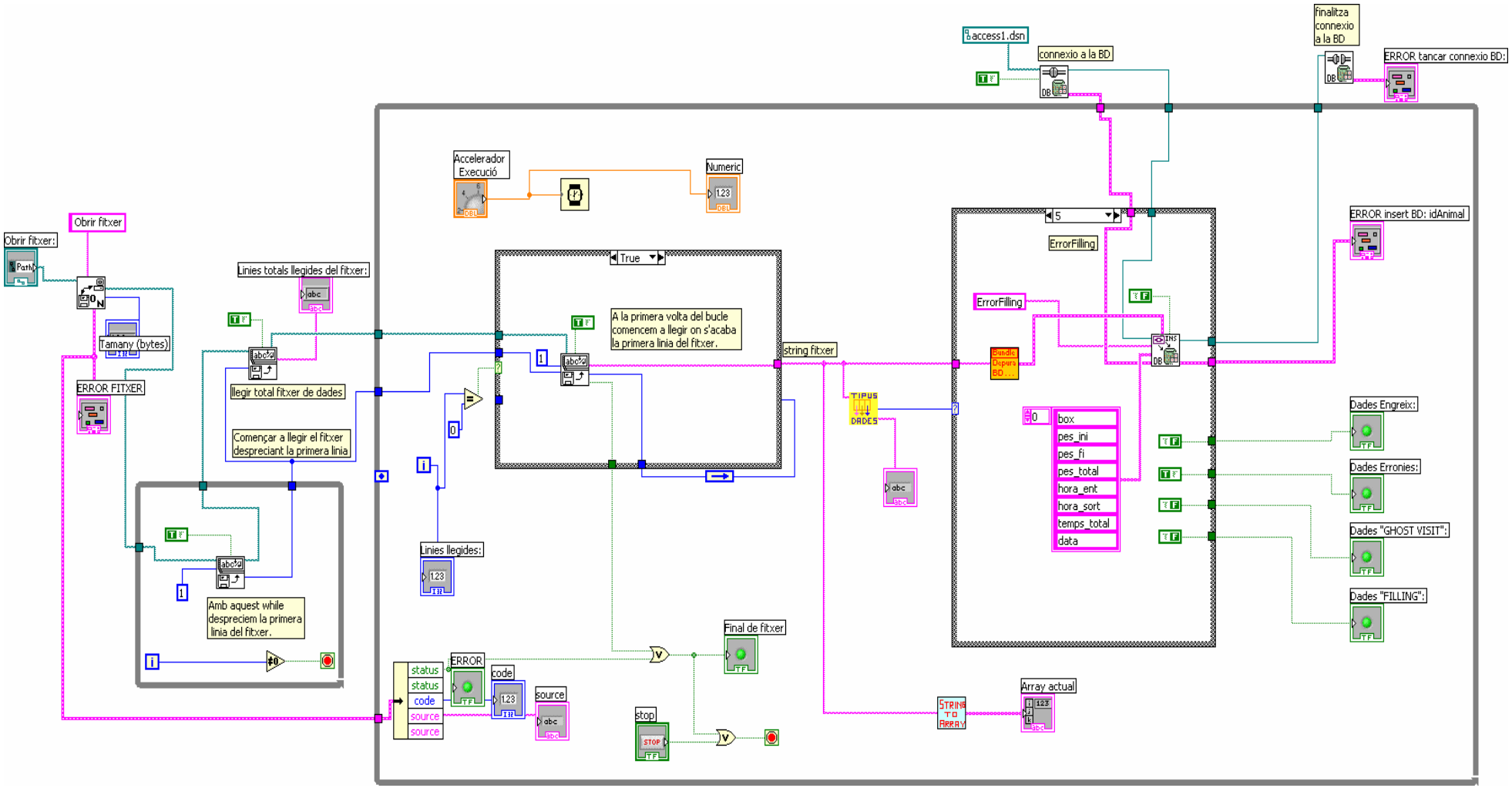
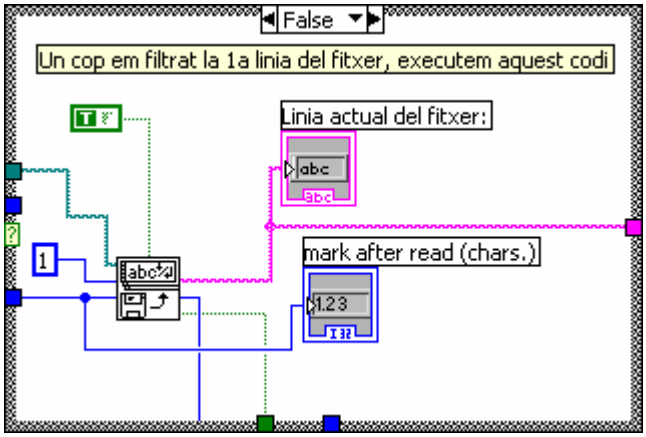


Figura 6.5-b: Diagram Block del programa *depuraBDdef.vi*.



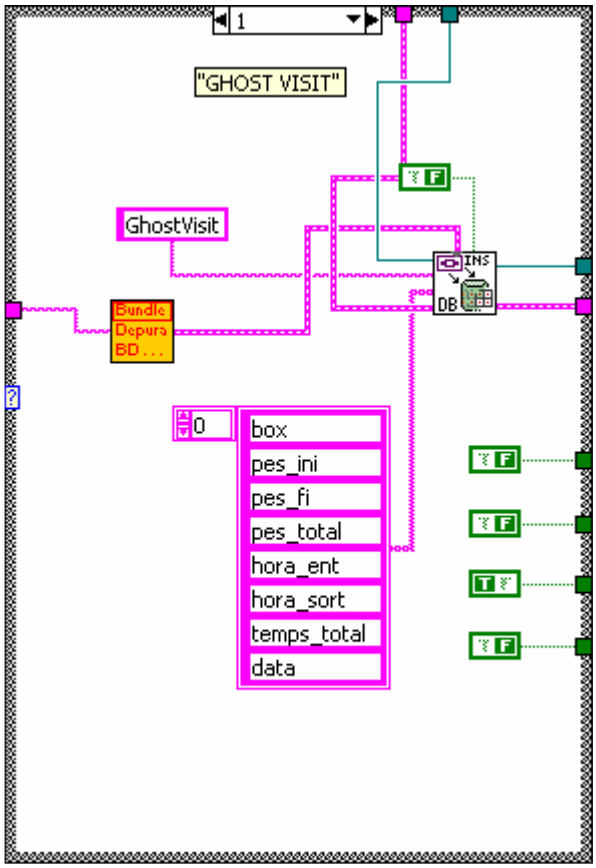


Figura 6.5-c: Diferents *CASES (condicionals)* del programa *depuraBDdef.vi.*

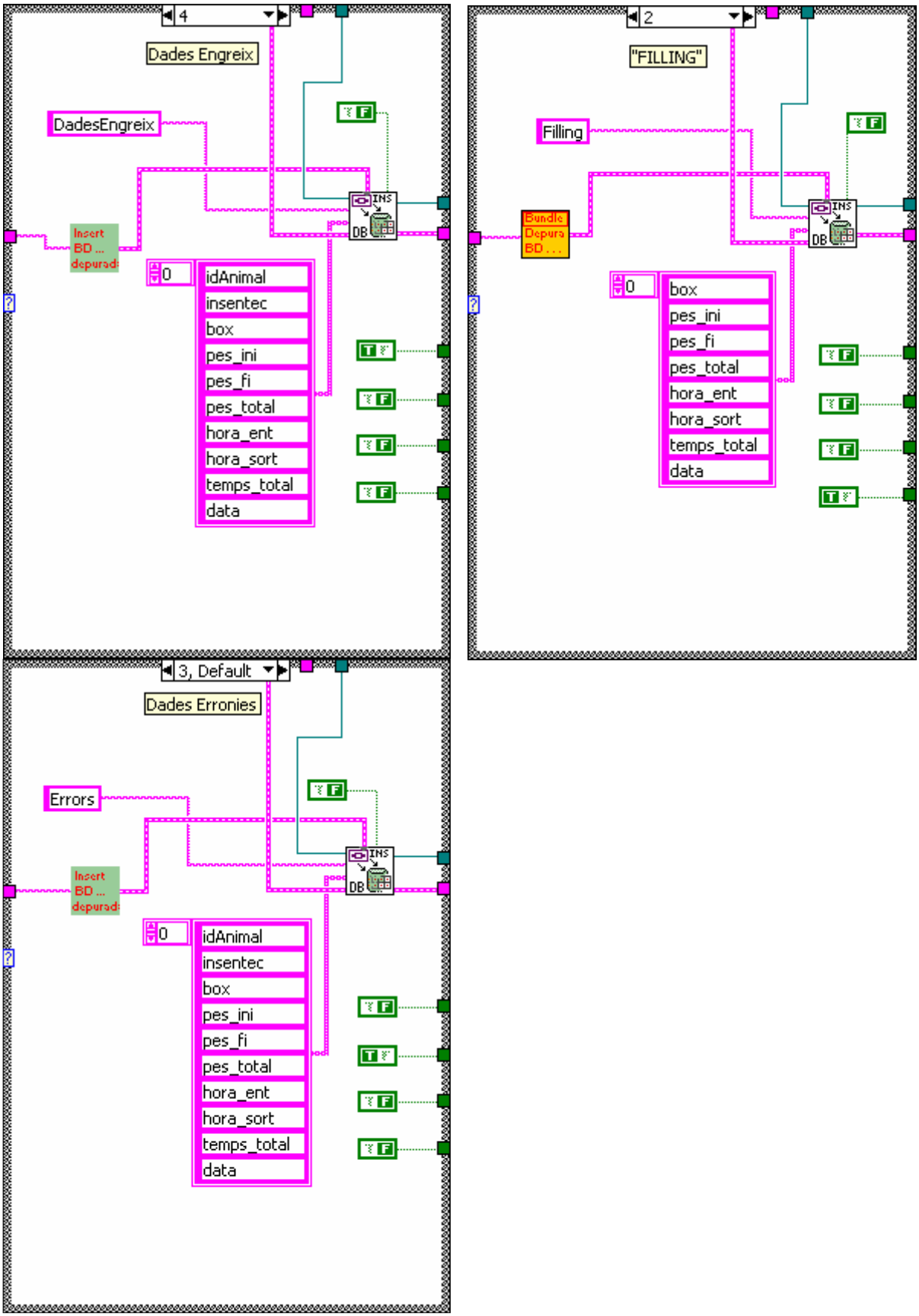


Figura 6.5-d: Diferents *CASES (condicionals)* del programa *depuraBDdef.vi.*

Llistat de subVIs del programa depuraBDdef.vi.

Read Lines From File.vi
Llegeix un fitxer de text línia per línia.

Open/Create/Replace File.vi
Obrir, crear o reemplaçar un fitxer.

DB Tools Open Connection.vi
Obrir la connexió a la BD.

DB Tools Close Connection.vi
Tancar la connexió a la BD.

subVI-depuraBD1.vi
Separar en camps de la BD els *strings* que arriben per inserir-los posteriorment a les taules ErrorFilling, GhostVisit i Filling.

DB Tools Insert Data.vi
Insertar dades a la BD.

subVI-insertBDdepurada.vi
Separar en camps de la BD els *strings* que arriben per inserir-los posteriorment a les taules DadesErronies i DadesEngreix.

subVI-tipusDades3.vi
Determinar de quin tipus són les dades a inserir a la BD.

subVI-stringToArray.vi
Transformar un *string* (cadena de caràcters) en un *Array* (matriu) per poder realitzar diferents operacions posteriors amb els valors de cadascun dels camps de la matriu.

subVI-depuraBD1.vi_____

Aquest subprograma el que fa es separar cada string de la línia actual que arriba del fitxer, en els camps corresponents a la BD per poder inserir-los posteriorment a cada columna de la taula corresponent.

Es molt similar al subprograma "subVI-insertBDdepurada" l'unic que aquest només inserirà dades que siguin "Filling", "GhostVisit" o “ErrorFilling”.

Connector Pane

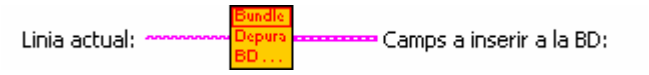


Figura 6.5-d: Connector Pane (icona amb connectors) del *subVI-depuraBD1.vi.*

Front Panel

Figura 6.6-a: Front Panel del subprograma *subVI-depuraBD1.vi.*.

Block Diagram

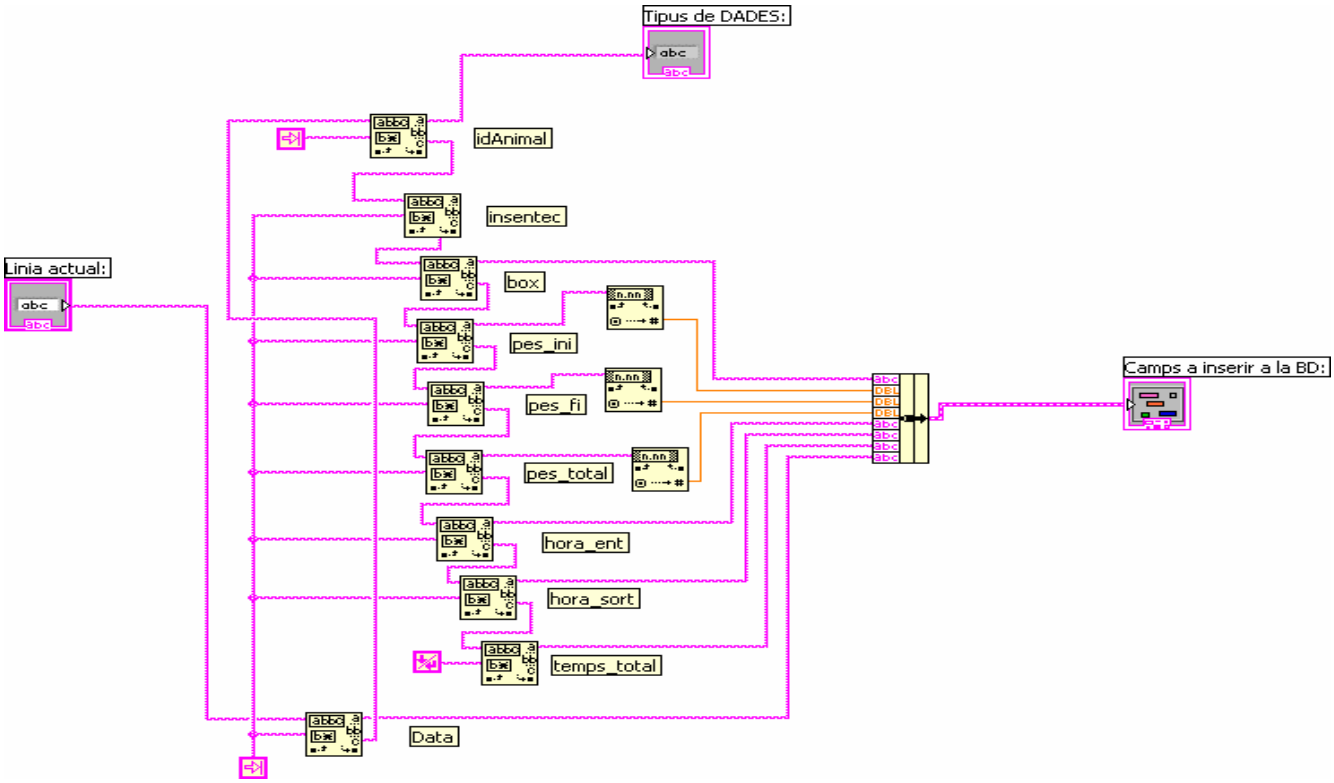


Figura 6.6-b: Block Diagram del subprograma *subVI-depuraBD1.vi.*.

subVI-insertBDdepurada.vi_____

Aquest subprograma el que fa és separar cada *string* de la línia actual que arriba del fitxer, en els camps corresponents a la BD per poder inserir-los posteriorment a cada columna de la taula corresponent.

Connector Pane

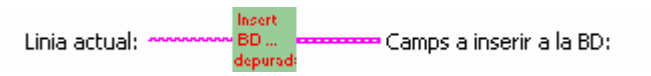


Figura 6.7-a: Connector Pane del subprograma *subVI-insertBDdepurada.vi.*

Front Panel

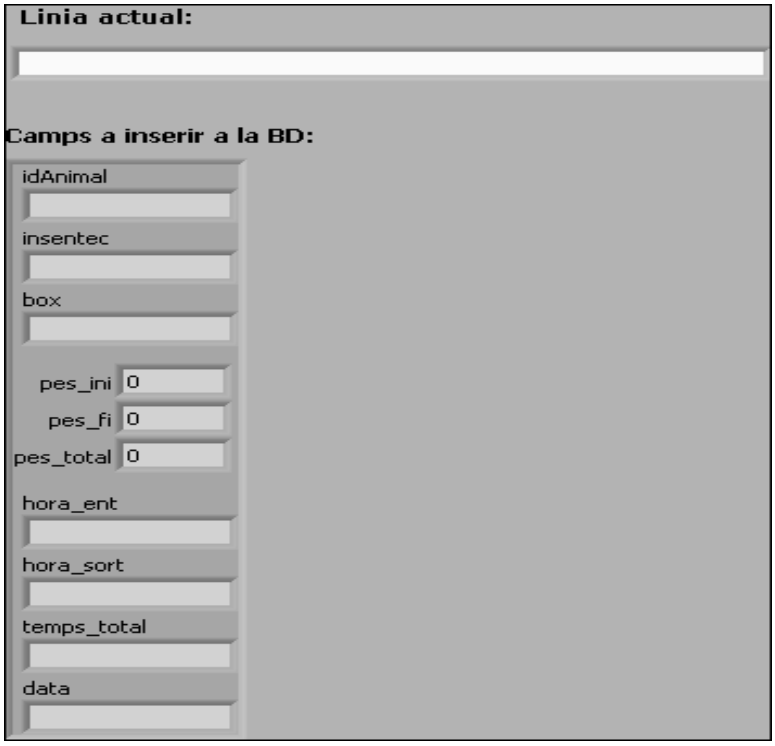


Figura 6.7-b: Front Panel del subprograma *subVI-insertBDdepurada.vi.*

Block Diagram

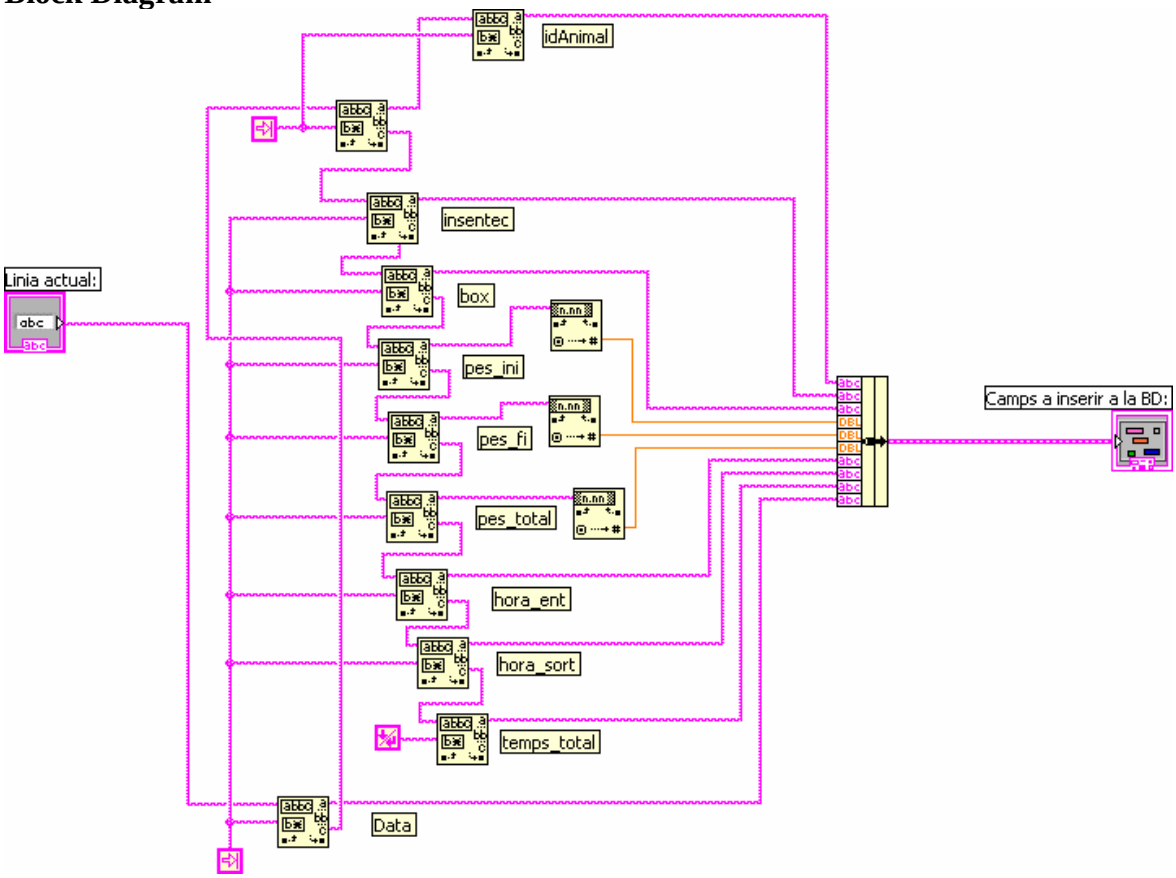


Figura 6.7-c: Block Diagram del subprograma *subVI-insertBDdepurada.vi.*

subVI-tipusDades3.vi_____

Aquest subprograma el que fa es rebre la línia actual i determinar de quin tipus és, per poder inserir-la posteriorment a la BD, sabent quin és el seu camp corresponent.

Connector Pane

Figura 6.8-a: Connector Pane del subprograma *subVI-tipusDades3.vi.*

Front Panel

Figura 6.8-b: Front Panel del subprograma *subVI-tipusDades3.vi.*

Block Diagram

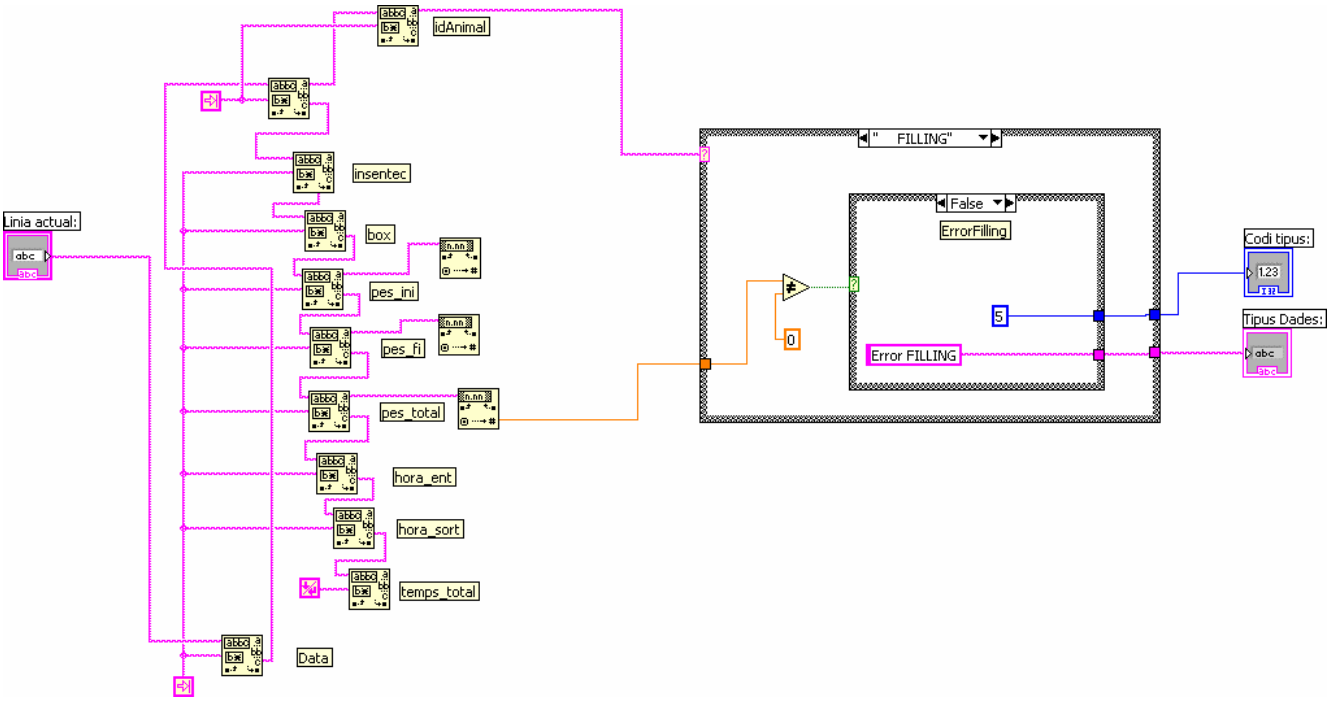


Figura 6.8-c: Block Diagram del subprograma *subVI-tipusDades3.vi.*

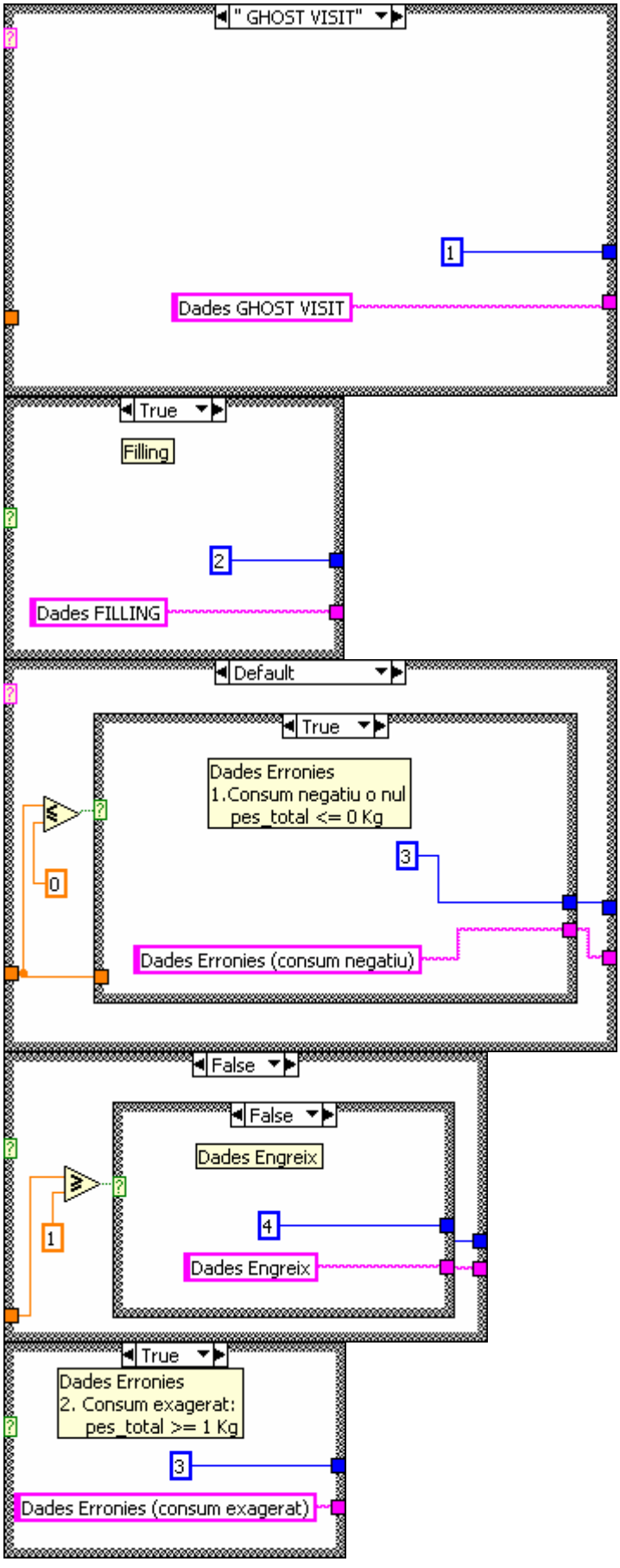


Figura 6.8-d: *CASES* (condicionals) del subprograma *subVI-tipusDades3.vi.*

subVI-stringToArray.vi

Connector Pane

Figura 6.8-a: Connector Pane del subprograma *subVI-stringToArray.vi.*

Front Panel

Figura 6.8-b: Front Panel del subprograma *subVI-stringToArray.vi.*

Block Diagram

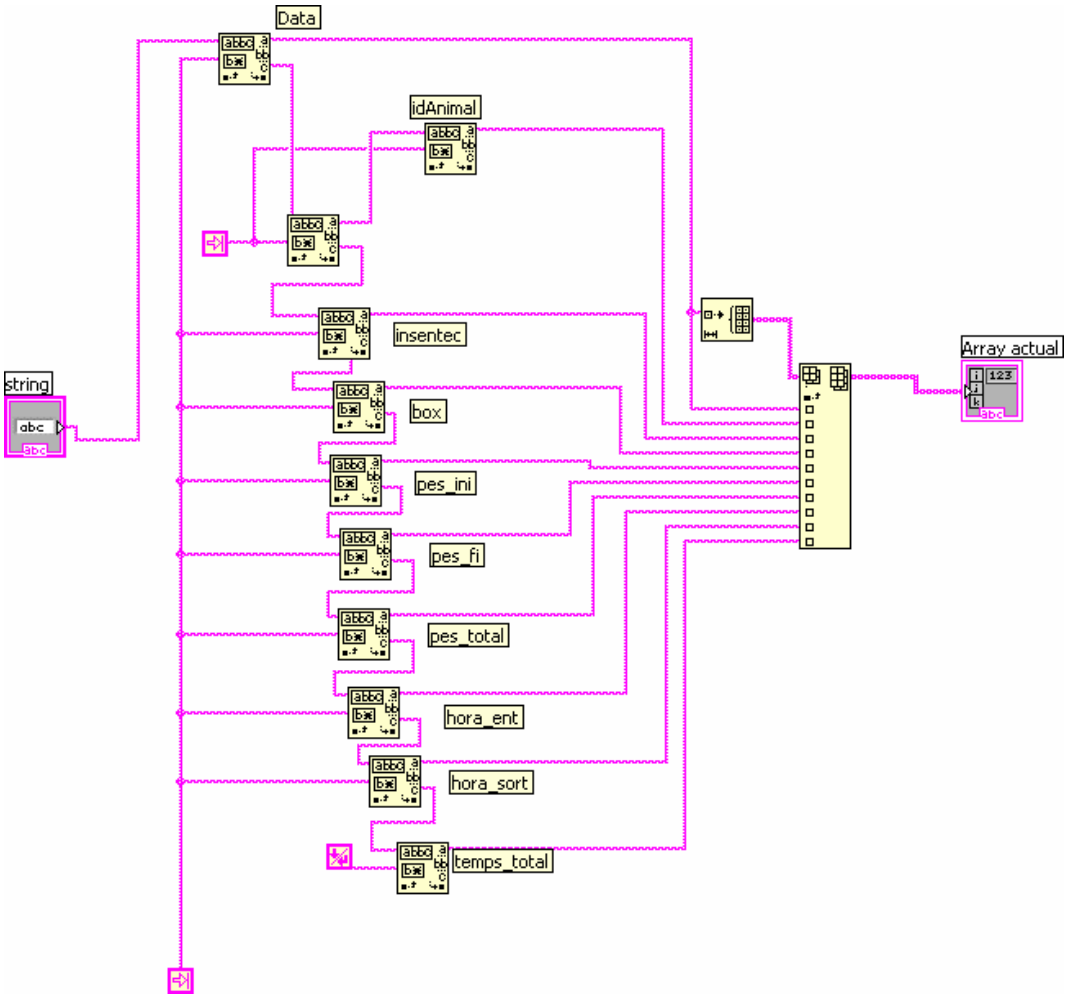


Figura 6.8-c: Block Diagram del subprograma *subVI-stringToArray.vi.*

6.2. Prototip de simulació amb *DataSocket.*

El prototip de simulació amb *DataSocket*, es desglossa en dos programes (*figura 6-D*) que a la vegada contenen varis sub-programes. La funcionalitat principal del prototip és la inserció telemàtica de les dades en una BD, simulant així el sistema d'alimentació automàtic, que rebrà un registre de dades cada cop que un animal entri a la menjadora i consumeixi certa quantitat de menjar en un moment determinat.

El primer programa envia les dades obtingudes d'un fitxer *.txt* línia per línia, utilitzant la tecnologia de *DataSocket* de *LabVIEW*. Això simula el que realitzaria els sensors instal·lats al sistema d'alimentació, quan enviessin les dades registrades per la entrada d'un animal. És en aquesta mateixa màquina que enviarà dades, on executarem el *DataSocket Server*, així doncs serà el pc servidor.

El segon programa rep les dades enviades per l'anterior programa, simulant doncs, el procés de rebuda de dades dins del sistema global, que li enviaran des d'altres programes, o de la mateixa menjadora. La màquina que rep les dades, és el pc client, que pot ser la mateixa màquina del client (local) o un altre pc remot, connectat sempre al mateix canal de *DataSocket* en que envia el servidor.

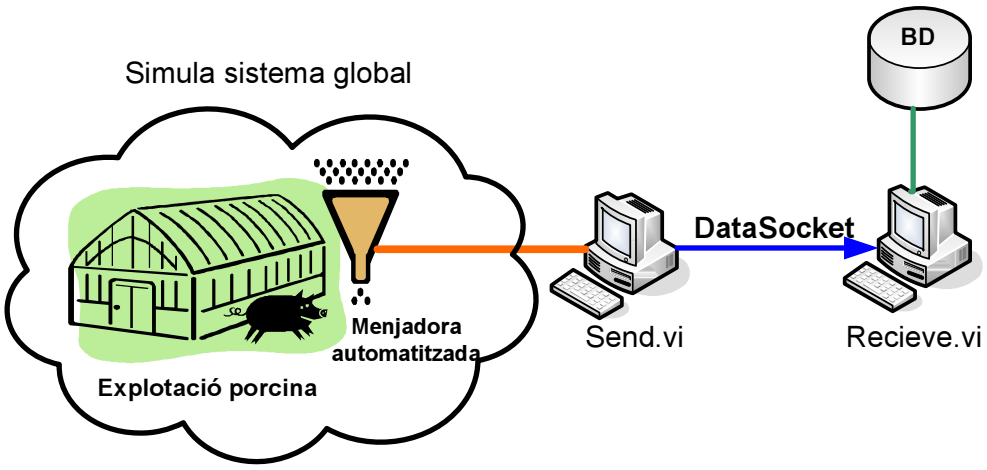


Figura 6-D: Diagrama de les eines de simulació d'enviament i rebuda de dades via *DS*.

6.2.1. Programa de enviament de dades.

send2.vi

Aquest programa el que pretén és captar les dades que es troben en un *fitxer.txt*, per poder-les enviar posteriorment (línia a línia) a un altre programa que tractarà aquestes dades.

En aquest programa utilitzarem al tecnologia *Data Socket Server* de *LabVIEW* per comunicar els programes i transferir les dades, fent de servidor aquest programa.

Connector Pane

Figura 6.9-a: Connector Pane (icono) del programa *send2.vi*.

Front Panel

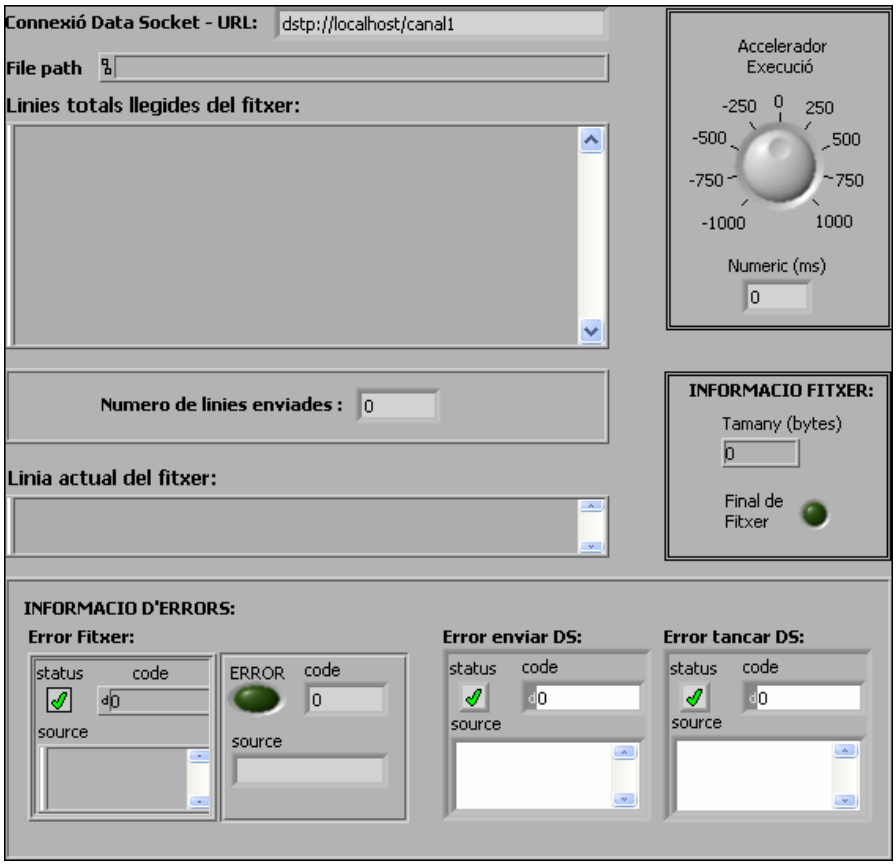


Figura 6.9-b: Front Panel del programa *send2.vi*.

Block Diagram

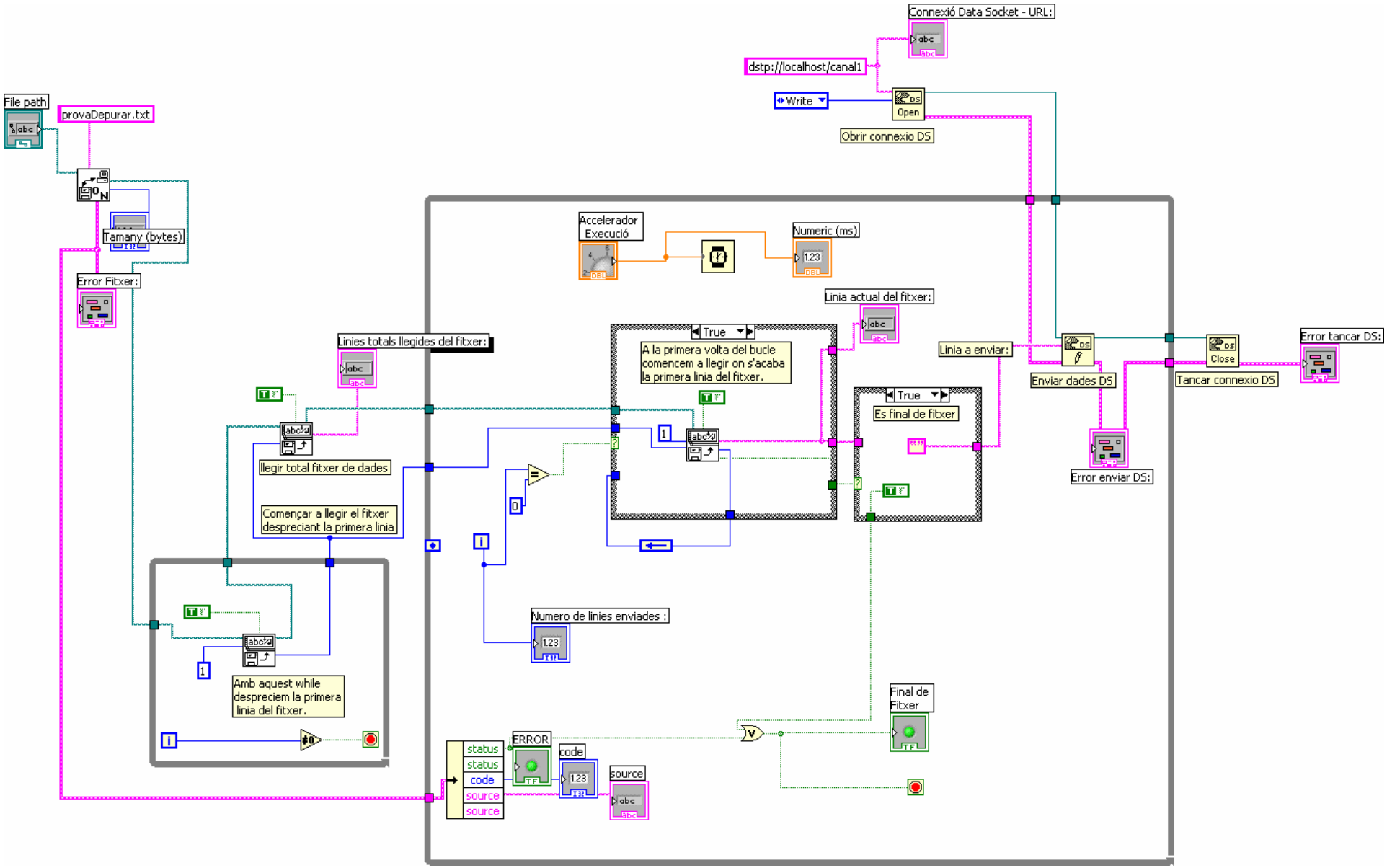


Figura 6.9-c: Block Diagram del programa *send2.vi.*

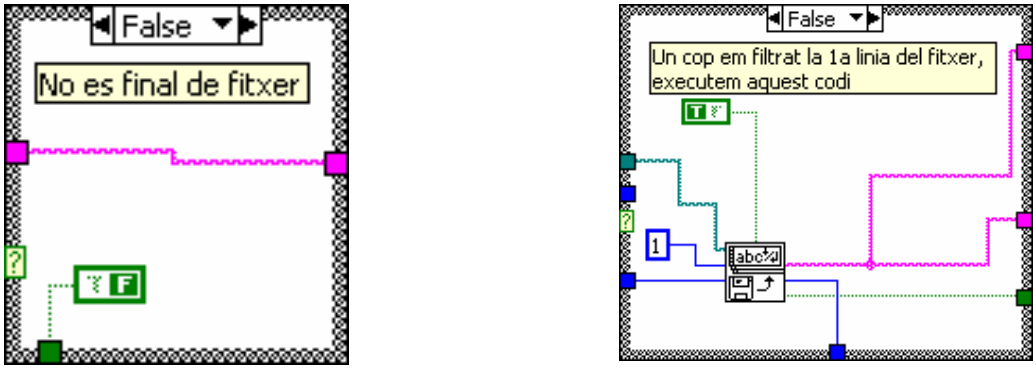


Figura 6.9-d: Diferents condicionals del programa *send2.vi.*

Position in Hierarchy

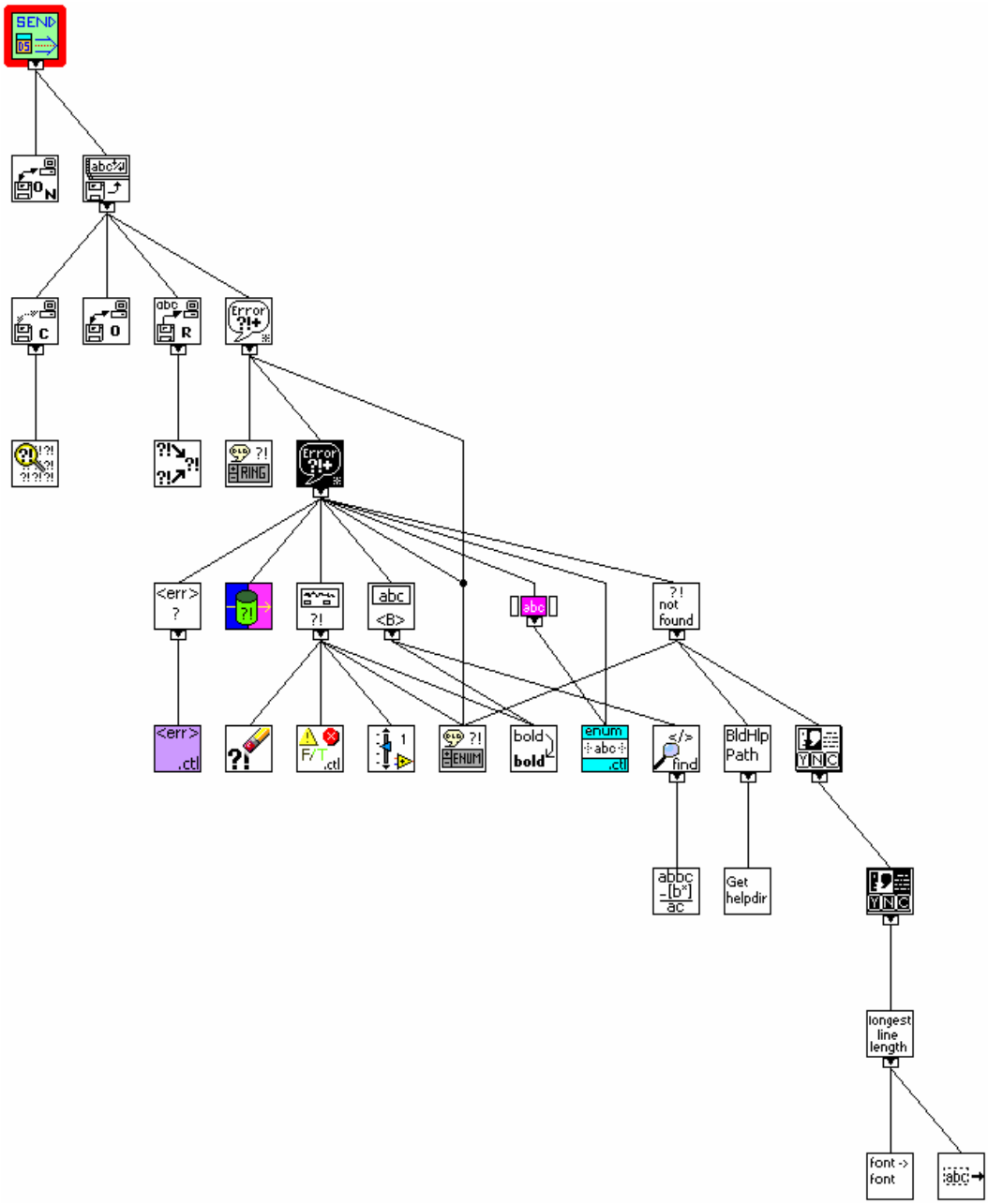


Figura 6.9-e: Jerarquia de tots els subprogrames que utilitza el programa *send2.vi.*

6.2.2. Programa de recepció i emmagatzemament de dades.

recieve1.vi

Aquest programa el que pretén és separar les dades que entren línia a línia (barrejades i amb possibles errors) procedents d'un altre programa i utilitzant la tecnologia de *DataSocket Server* de *LabVIEW*, per insertar-les posteriorment a una BD depurada dividida en 5 taules:

1. DadesEngreix
   (conjunt de dades rellevants dels animals)

2. Filling
   (dades que es recullen després de cada omplida de les menjadores)

3. GhostVisit
   (per enregistrar les visites fantasmes)

4. Errors
   (conte el conjunt de les dades errònies i irrellevants dels animals)

5. ErrorFilling
   (conte els errors de les dades Filling)

Connector Pane

Figura 6.10-a: Connector Pane (icono) del programa *recieve1.vi*.

Front Panel

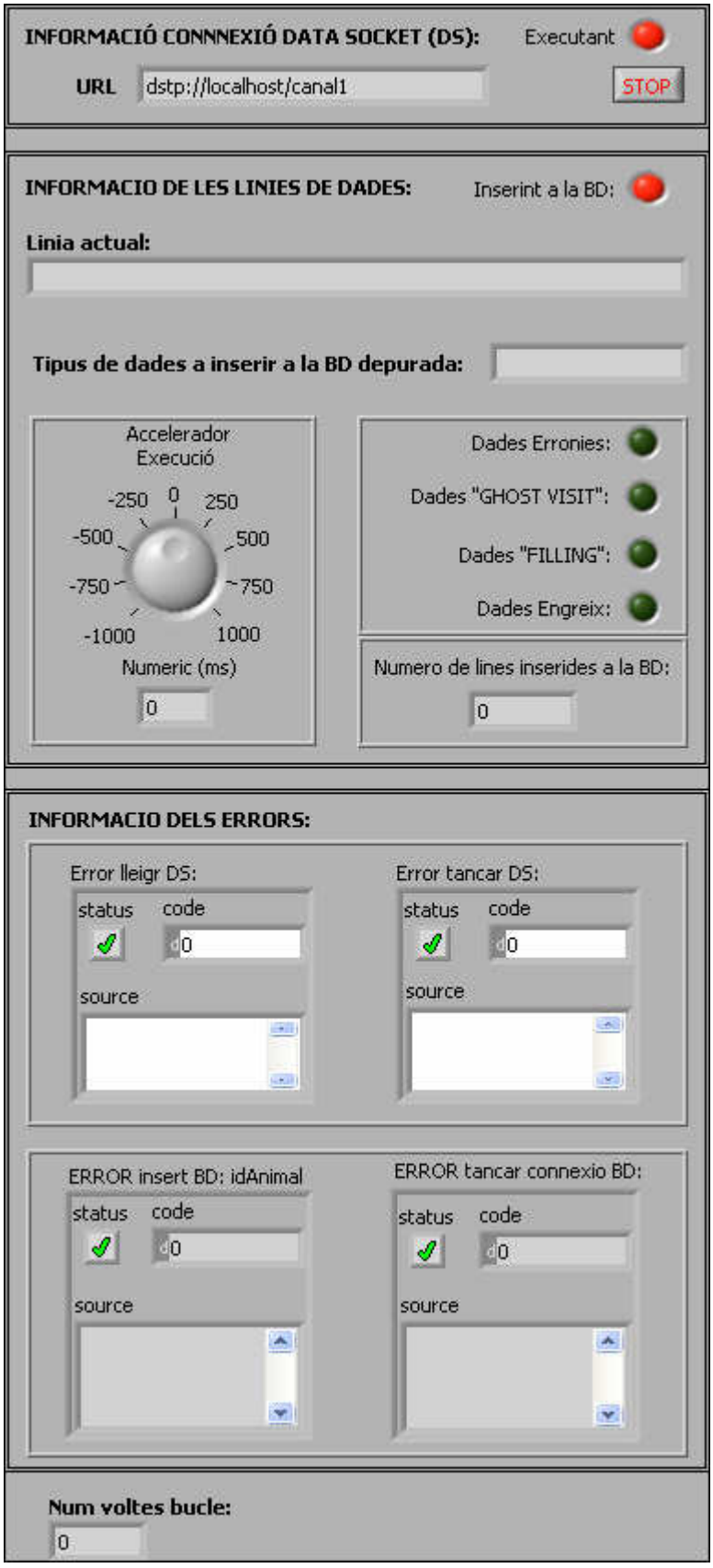


Figura 6.10-b: Front Panel del programa *recieve1.vi.*

Block Diagram

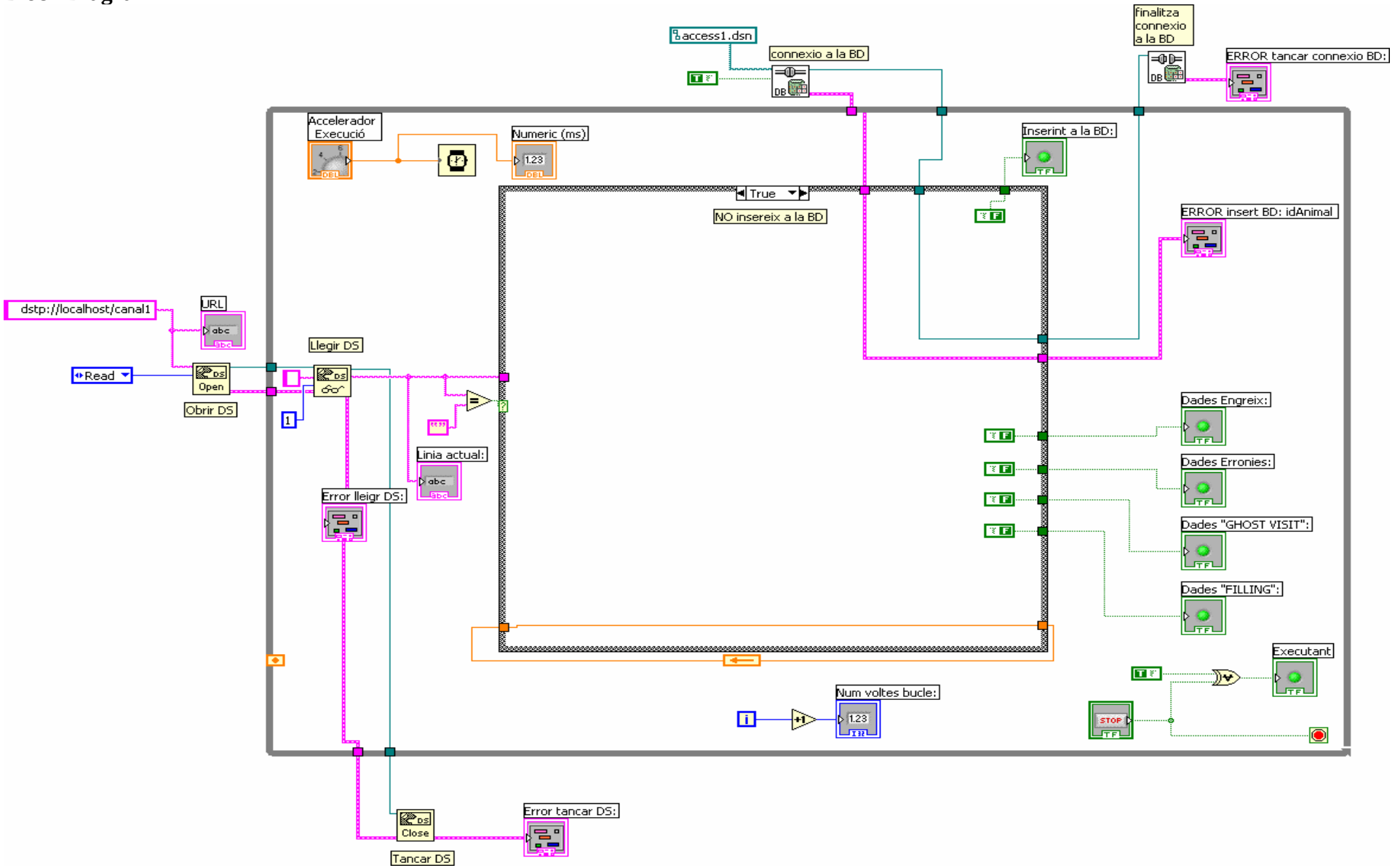


Figura 6.10-c: Block Diagram del programa *recieve1.vi*

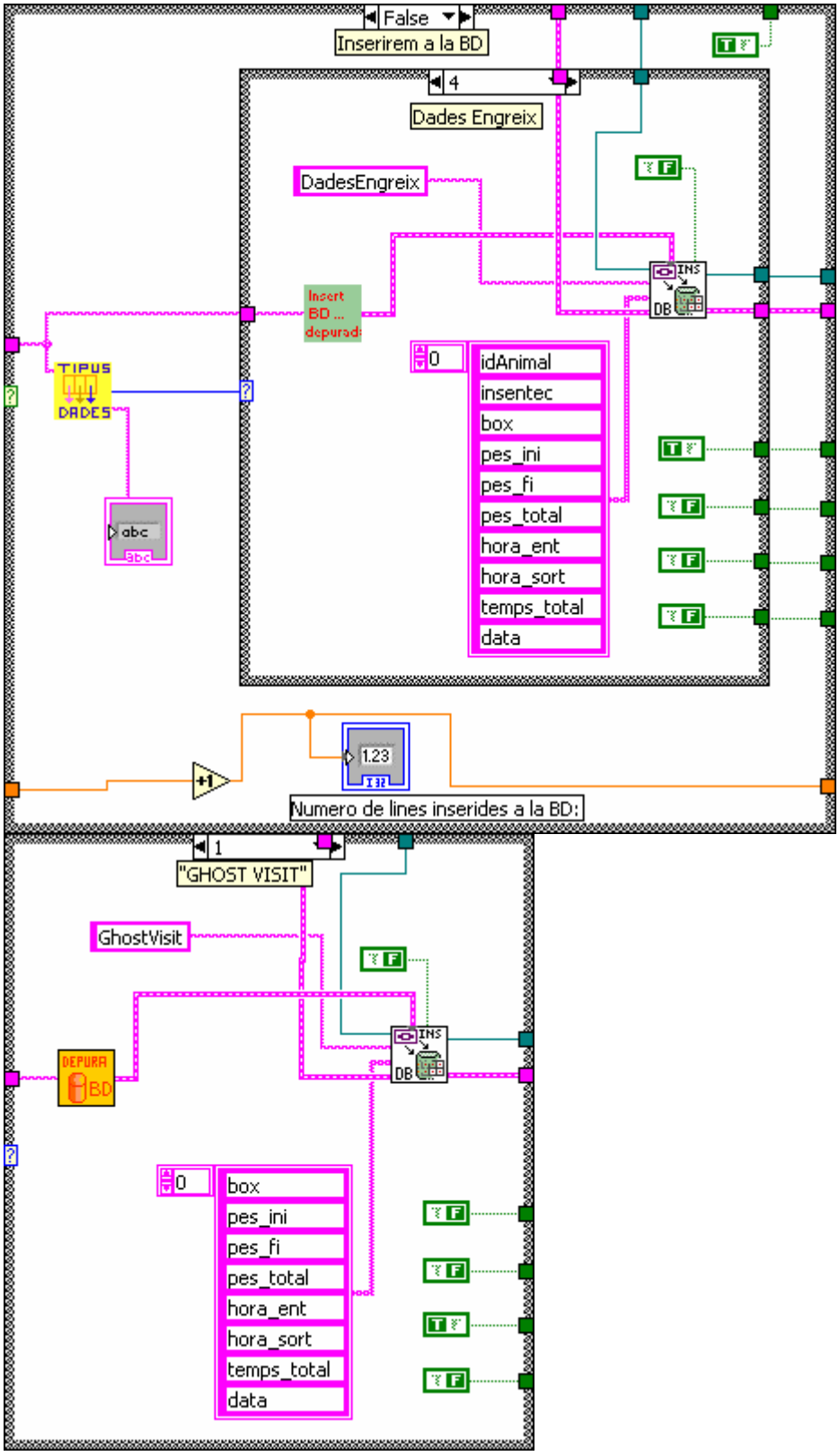


Figura 6.10-d: Diferents *CASES* (condicionals) del programa *recieve1.vi*

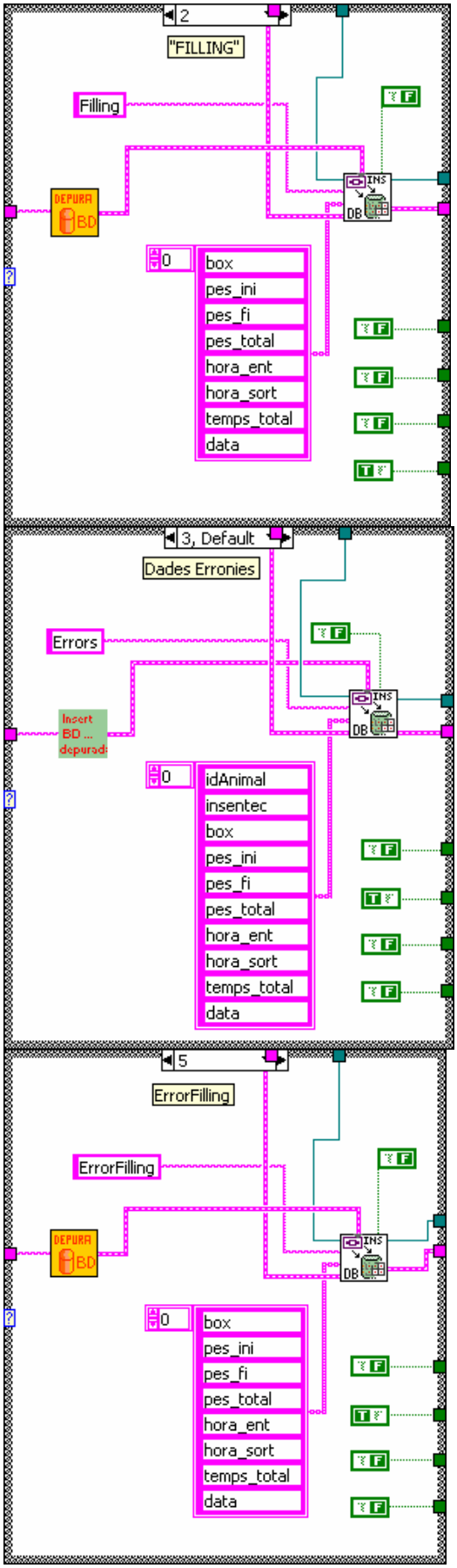


Figura 6.10-e: Diferents *CASES* (condicionals) del programa *recieve1.vi*

6.3. Prototip de consulta a la BD.

A continuació mostrarem les implementacions del prototip de consulta a les bases de dades (*figura 6-C*). Aquest prototip consta de dues parts: consulta de dades localment i consulta de dades remotament utilitzant el *WebServer* de *LabVIEW (figura 6.0-c)*.

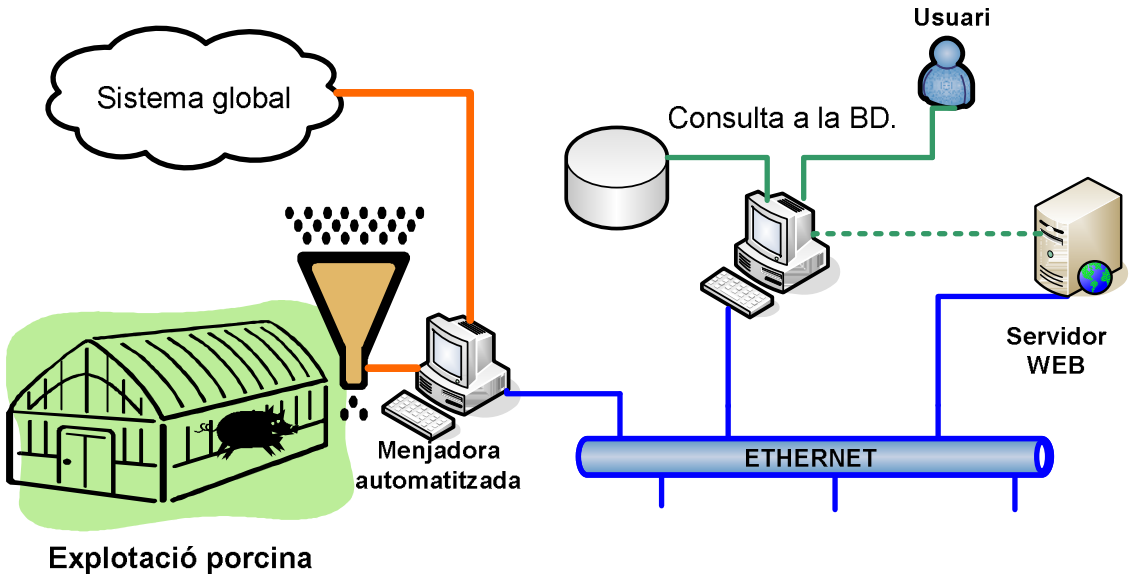


Figura 6-C: Diagrama de les eines de consulta a la BD local i remota via web.

6.3.1. Consultar a la base de dades localment.

consulta2-1-3.vi_____

Aquest programa realitzarà consultes a una base de dades predeterminada.
Les consultes es faran amb SQL, sabent la estructura de la BD, es a dir, les taules, els camps de cada taula, el tipus ...

Exemple de consulta: (pot anar tot a la mateixa linia i no cal acabar amb ";")

```
Select *
From DadesEngreix
Where box=3 and pes_total>0.2
```

Les taules de la BD depurada son: *DadesEngreix, Filling, GhostVisit, Errors i ErrorFilling.*

Nota:
(El nom de les taules de la BD i dels seus camps han d'escriure's tal com estan a la BD).

Per reiniciar el programa i tornar a elegir una BD diferent a la que està connectada actualment, clicar al botó **RESET**.

Per parar el programa fer click a **STOP**.

Connector Pane

Figura 6.11-a: Connector Pane (icono) del programa *consulta2-1-3.vi*.

Front Panel

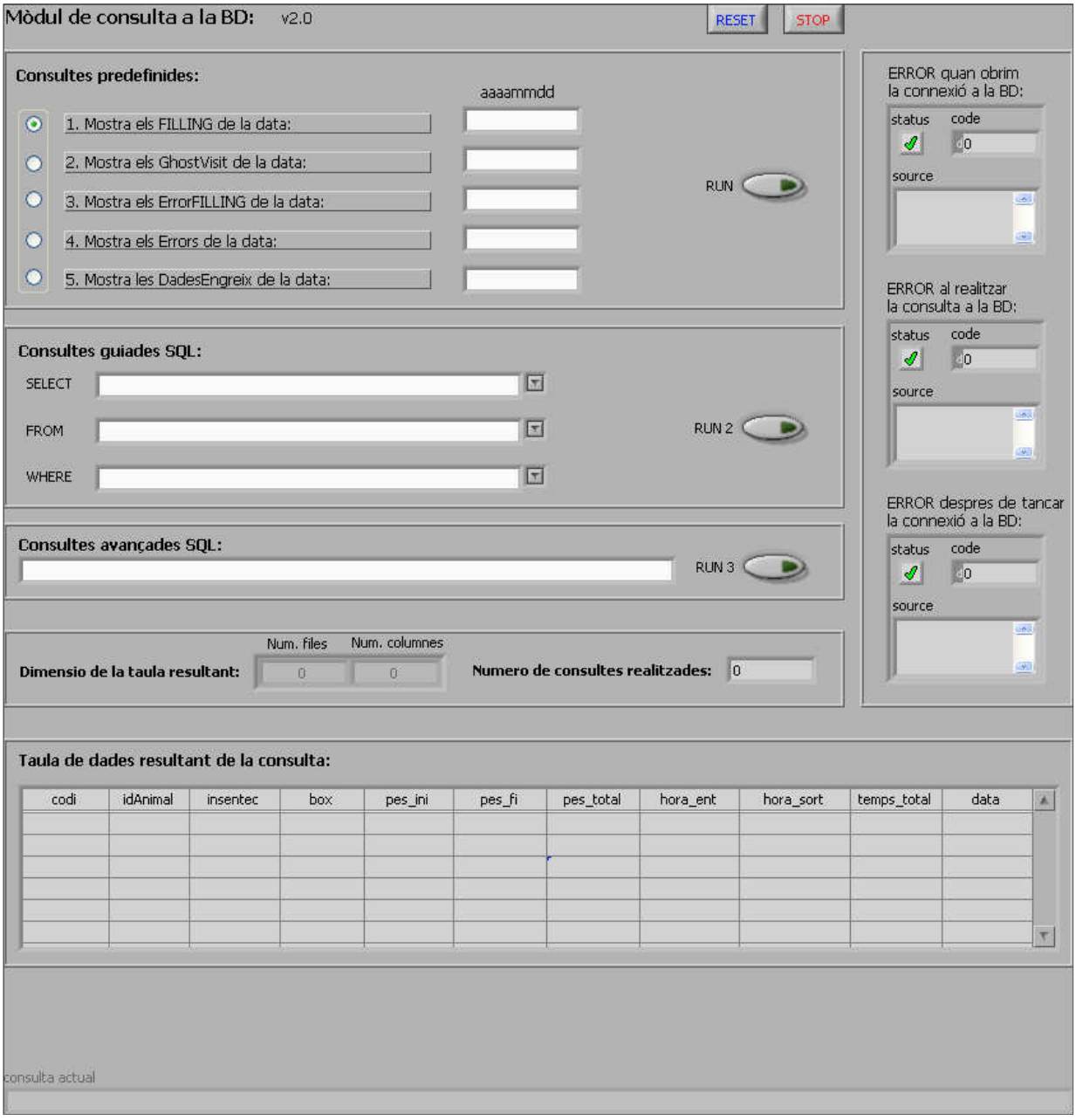


Figura 6.11-b: Front Panel (interfície) del programa *consulta2-1-3.vi.*

Block Diagram

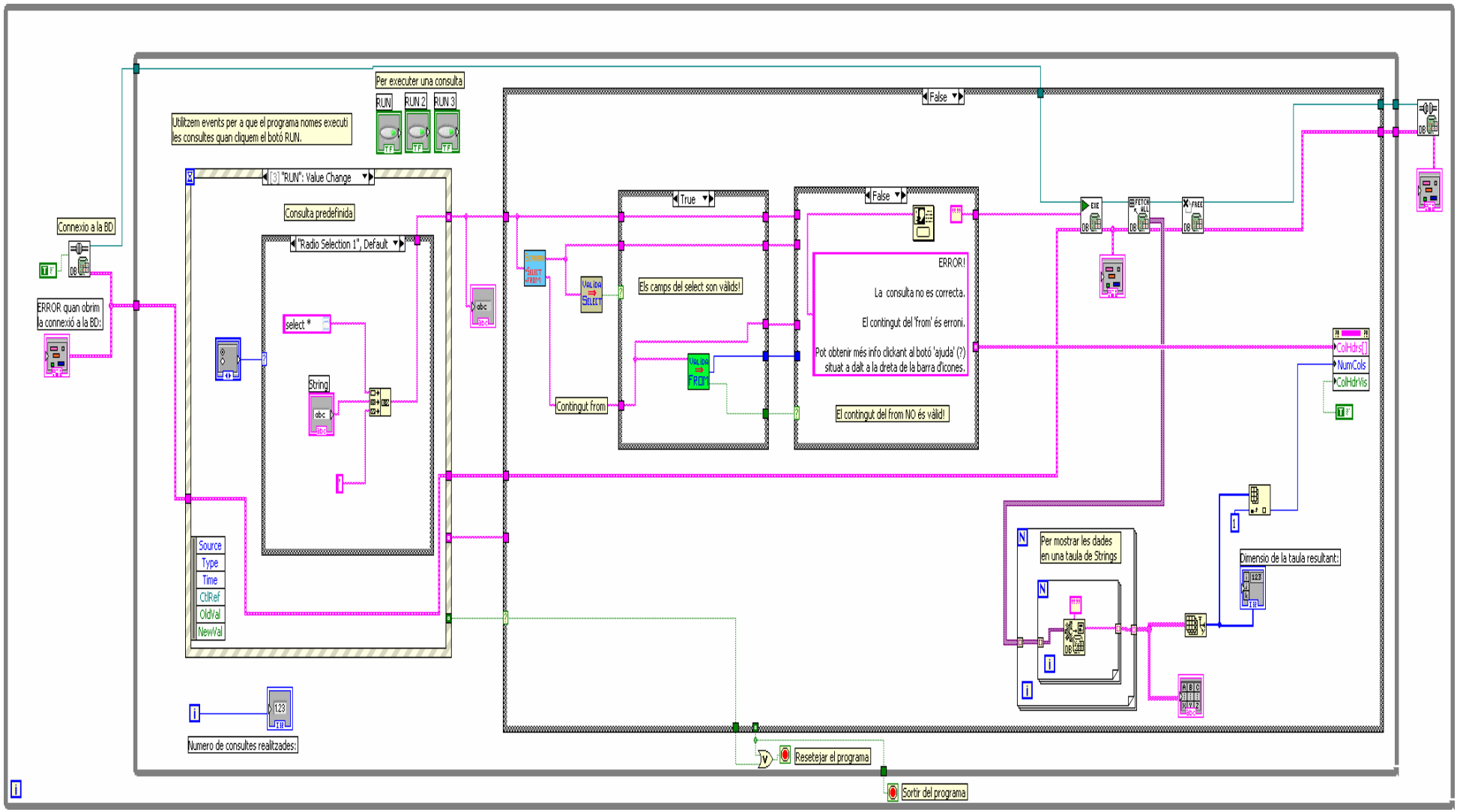


Figura 6.11-c: Block Diagram del programa *consulta2-1-3.vi.*

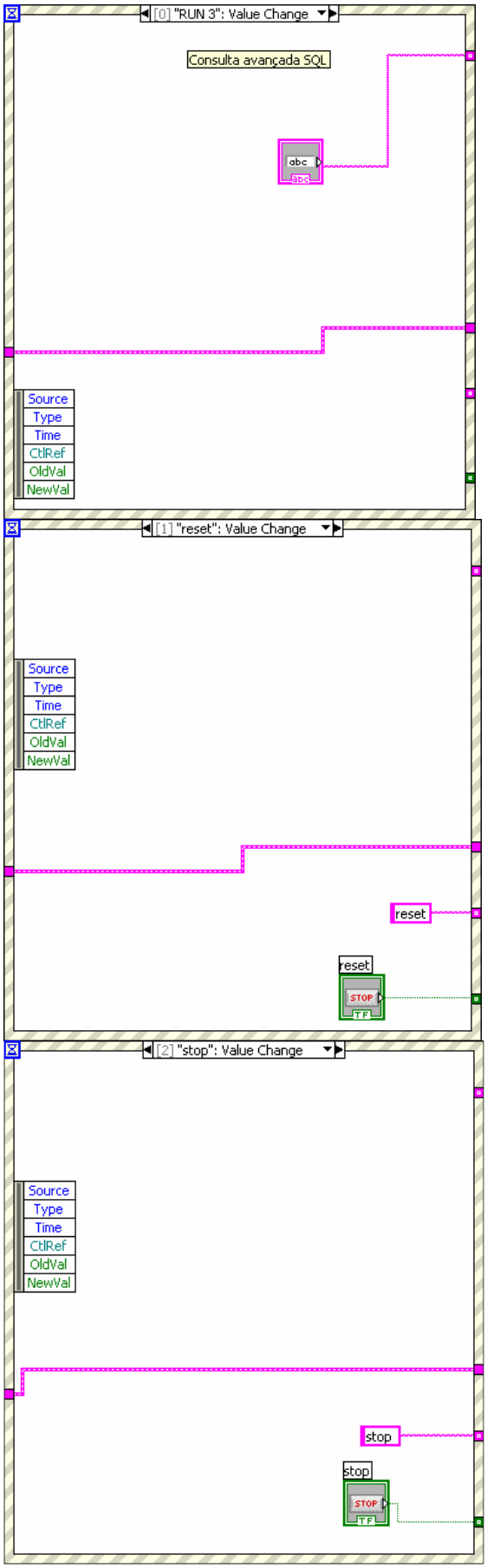


Figura 6.11-d: Diferents *Event Cases* del programa *consulta2-1-3.vi.*

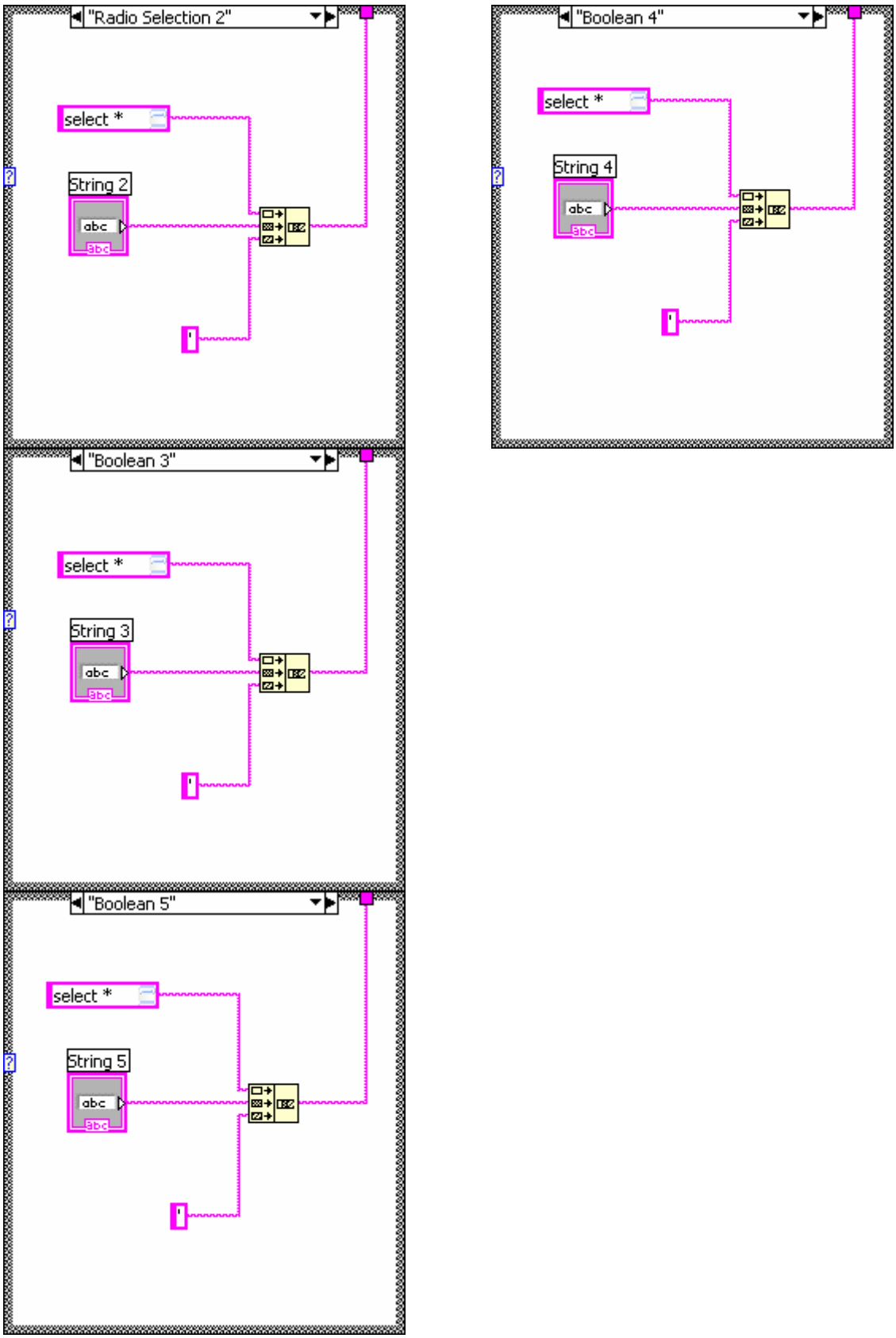


Figura 6.11-e: Diferents *Cases* (condicionals) del programa *consulta2-1-3.vi.*

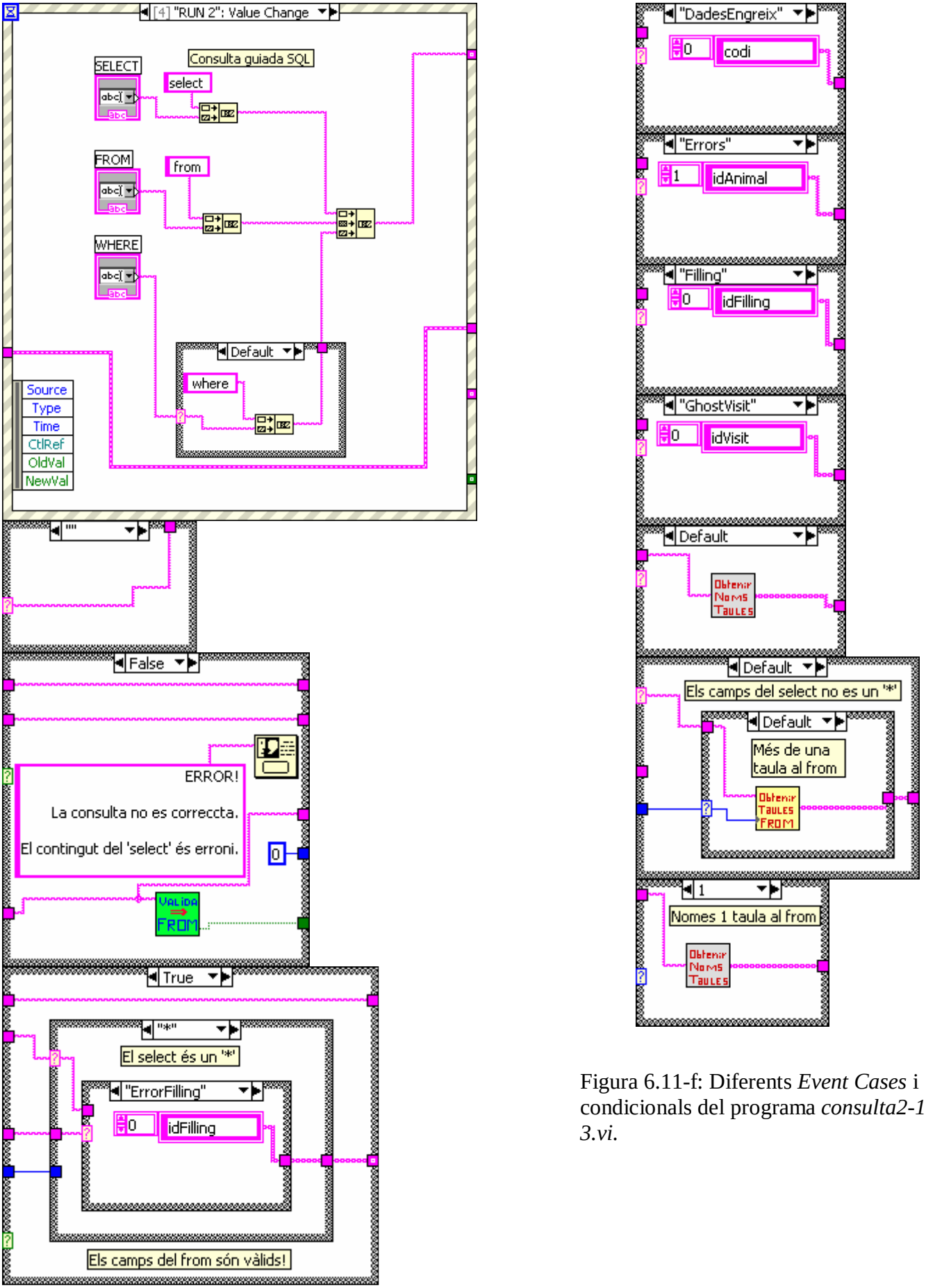


Figura 6.11-f: Diferents *Event Cases* i condicionals del programa *consulta2-1-3.vi*.

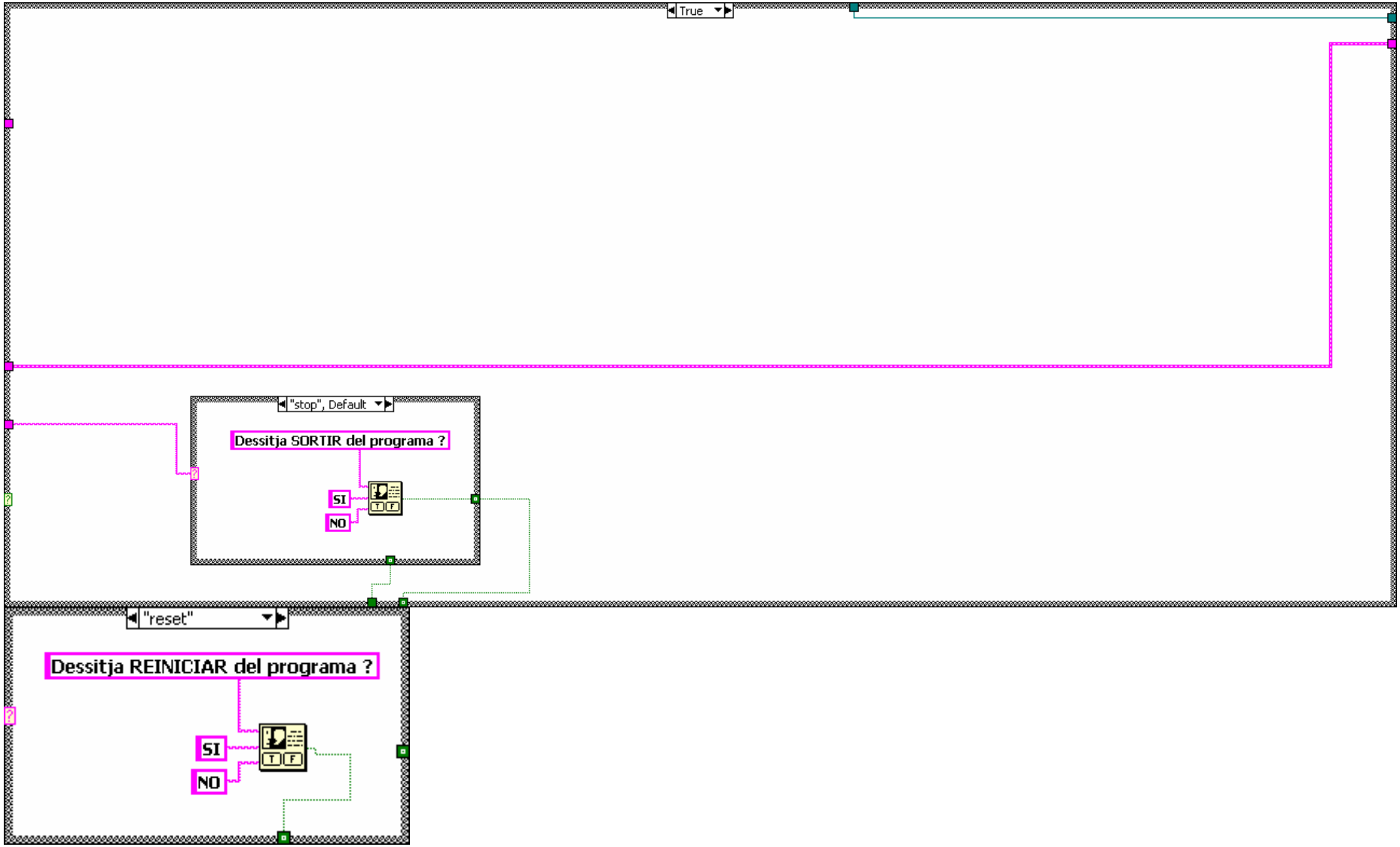


Figura 6.11-g: Diferents *Cases* (condicionals) del programa *consulta2-1-3.vi*.

Llista dels subVIs del programa *consulta2-1-3.vi.*

DB Tools Close Connection.vi
Tancar connexió amb la BD.

DB Tools Execute Query.vi
Executar consulta a la BD.

DB Tools Fetch Recordset Data.vi
Extreure les dades obtingudes de una consulta en una *array* (matriu) 2D.

Conn Execute.vi
Executar una consulta SQL y passa a fora una referència.

DB Tools Free Object.vi
Alliberar el objecte de la consulta destruint la referència associada.

sub-vi obtenirNomTaules.vi
Obtenir el nom de la taula de la BD on es vol realitzar la consulta.

sub-vi separarConsulta.vi
Separar el contingut del *select* i el contigut del *from* per comprovar que son camps vàlds en la nostra BD.

sub-vi validarFrom.vi
Validar que el contingut del *from* és vàlid per la nostra BD.

sub-vi validarSelect.vi
Validar que el contingut del *select* és vàlid per la nostra BD.

sub-vi obtenirTaulesDelFrom.vi
Insertar el nom de les taules resultants de la consulta SQL, per ficar-los en un array (matriu) per poder després ficar els títols a la taula resultant.

DB Tools Open Connection.vi
Obrir la connexió amb la BD.

sub-vi obtenirNomTaules.vi

Aquest sub-vi insertarà els noms que han de sortir en la taula resultant de la consulta, en un *array* (matriu) per després poder tractar-lo millor al programa principal de consulta a la BD.

Arribarà un *string* amb els noms separats per comes (serà el contingut del *select*) i sortirà un *array* de *strings* amb un nom per cel·la.

Connector Pane

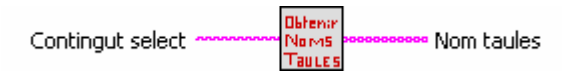


Figura 6.12-a: Connector Pane (icono amb connectors) del subprograma *sub-vi obtenirNomTaules.vi.*

Front Panel

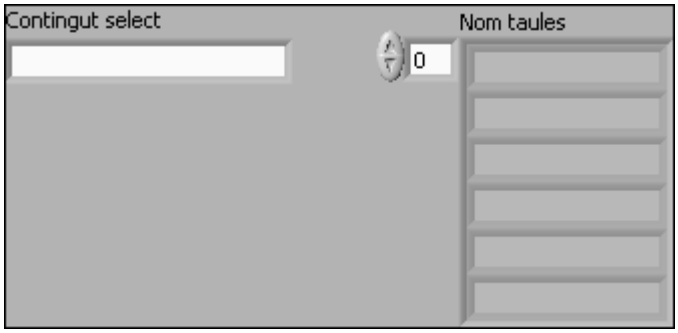


Figura 6.12-b: Front Panel del subprograma *sub-vi obtenirNomTaules.vi*.

Block Diagram

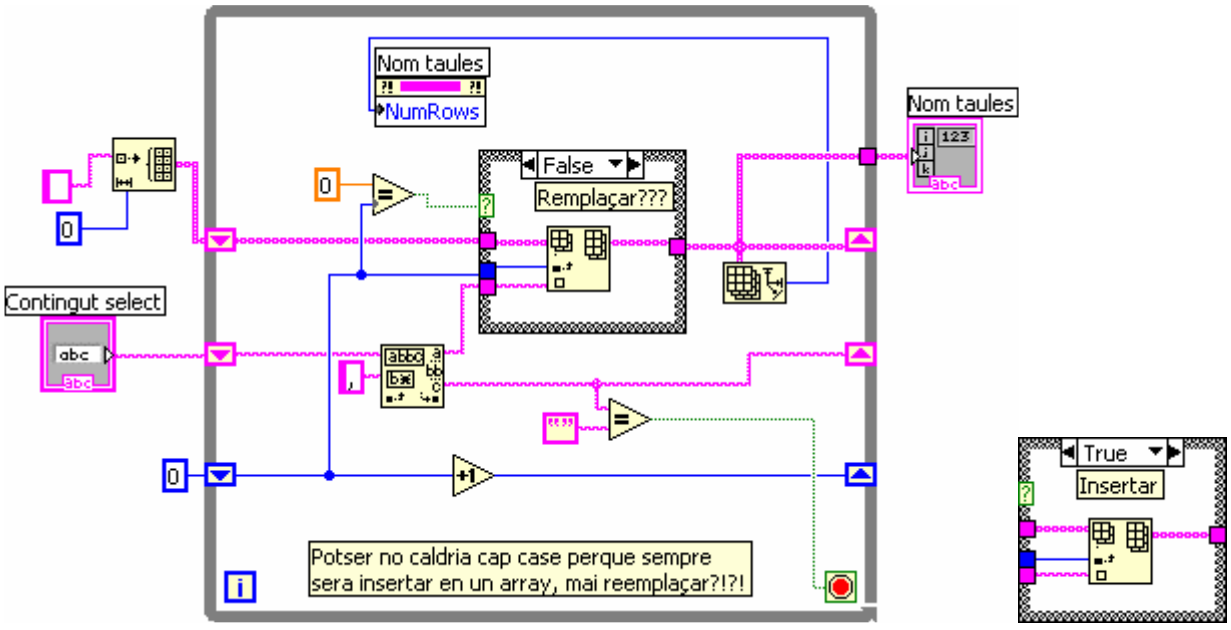


Figura 6.12-c: Block Diagram del subprograma *sub-vi obtenirNomTaules.vi.*

sub-vi separarConsulta.vi

Aquest subprograma separa el contingut del *select* i el contingut del *from* de la consulta realitzada.

Connector Pane

Figura 6.13-a: Connector Pane del subprograma *sub-vi separarConsulta.vi.*

Front Panel

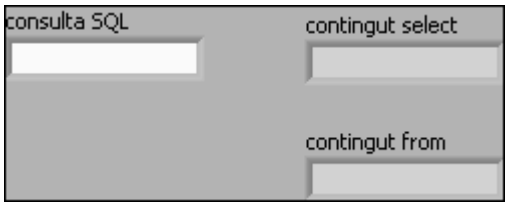


Figura 6.13-b: Front Panel del subprograma *sub-vi separarConsulta.vi*

Block Diagram

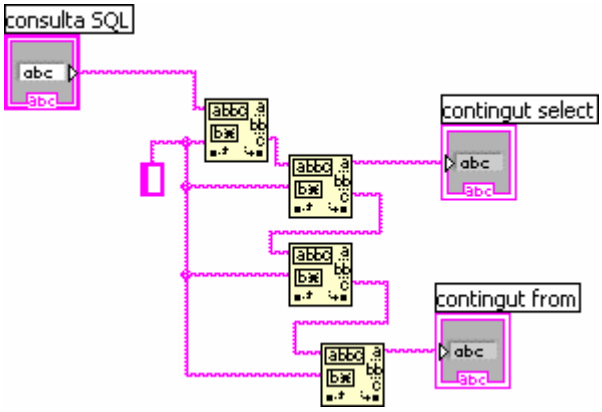


Figura 6.13-c: Block Diagram del subprograma *sub-vi separarConsulta.vi*

sub-vi validarFrom.vi

Aquest sub-vi validarà el contingut del *from* de la consulta.

Arribarà un string amb els camps separats per comes (serà el contingut del *from*) i sortirà un booleà indicant si els camps introduïts a la consulta són correctes o no.

Connector Pane

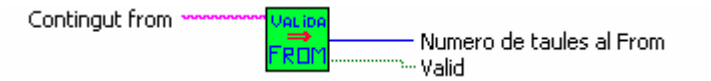


Figura 6.14-a: Connector Pane del subprograma *sub-vi validaFrom.vi.*

Front Panel

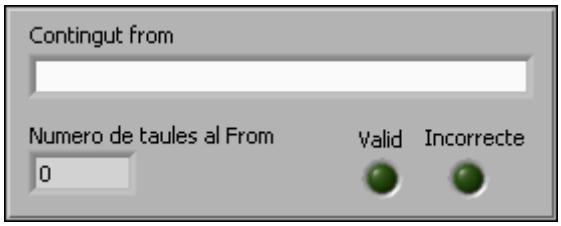


Figura 6.14-b: Front Panel del subprograma *sub-vi validaFrom.vi.*

Block Diagram

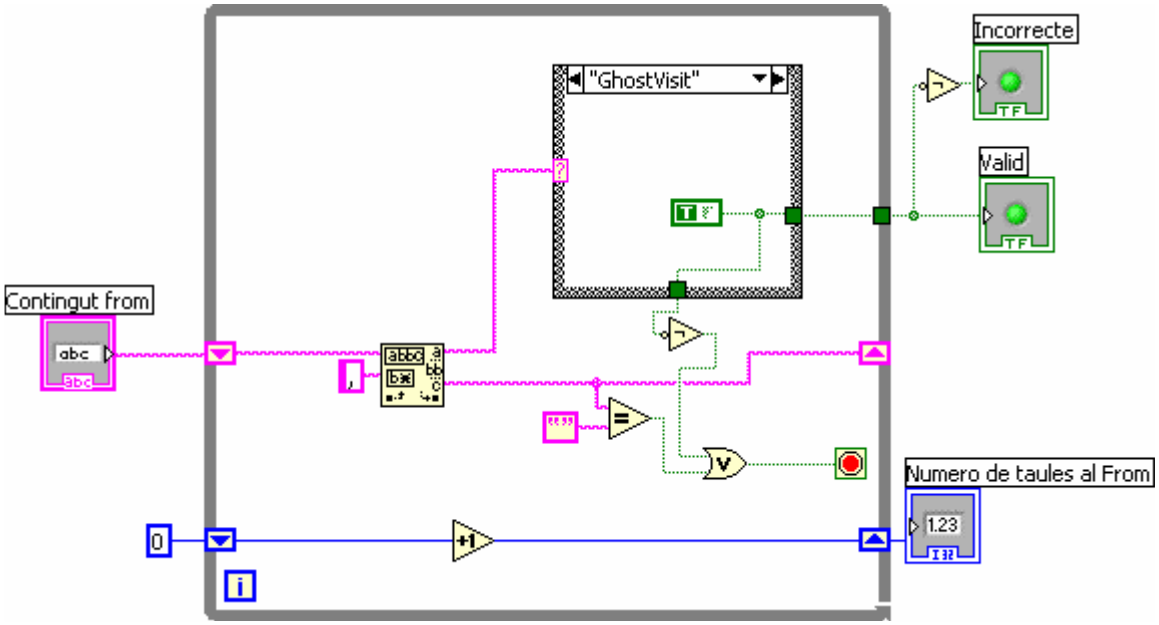


Figura 6.14-c: Block Diagram del subprograma *sub-vi validaFrom.vi.*

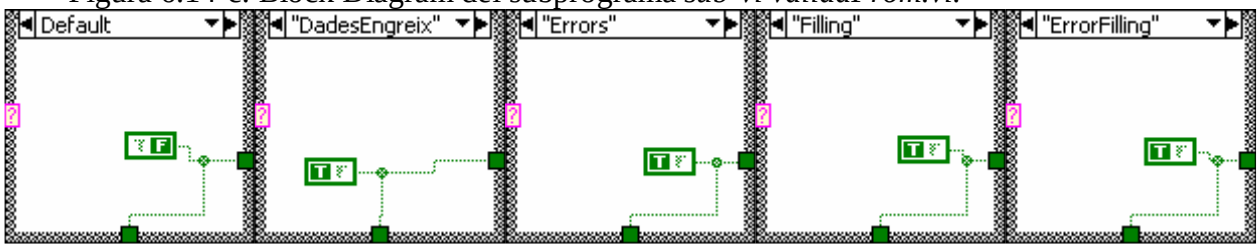


Figura 6.14-d: Condicionals del subprograma *sub-vi validaFrom.vi.*

sub-vi validarSelect.vi

Aquest sub-vi validarà el contingut del *select* de la consulta.

Arribarà un *string* amb els camps separats per comes (serà el contingut del *select*) i sortirà un booleà indicant si els camps introduïts a la consulta són correctes o no.

Connector Pane

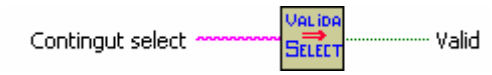


Figura 6.15-a: Connector Pane del subprograma *sub-vi validaSelect.vi.*

Front Panel

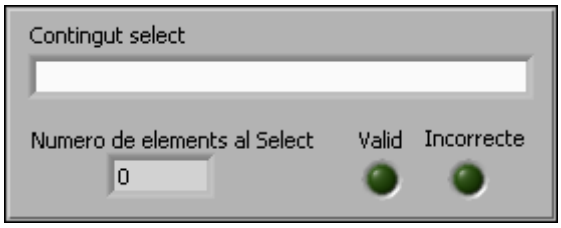


Figura 6.15-b: Front Panel del subprograma *sub-vi validaSelect.vi.*

Block Diagram

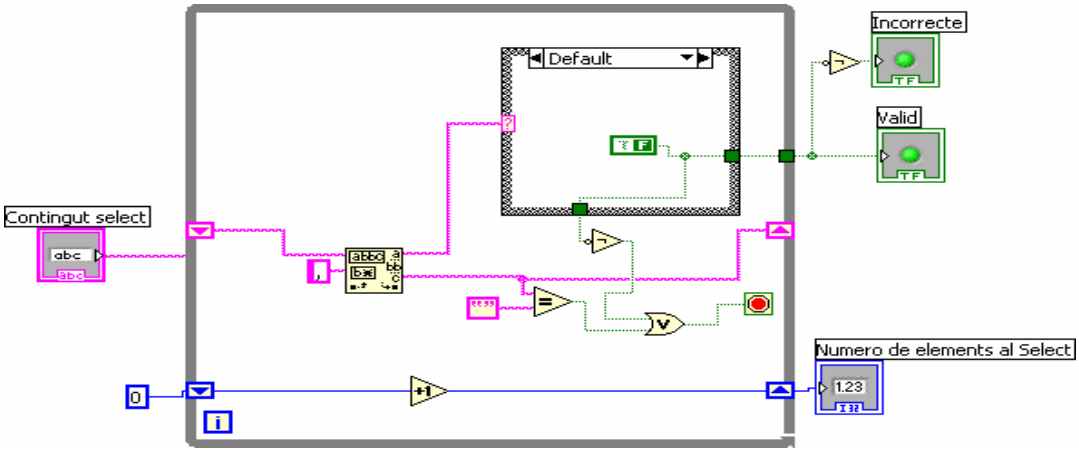
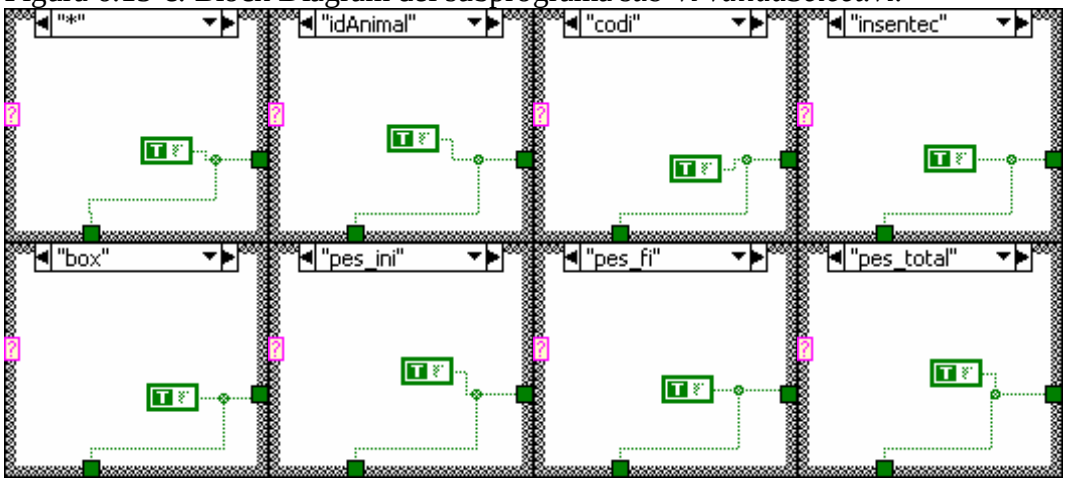


Figura 6.15-c: Block Diagram del subprograma *sub-vi validaSelect.vi.*



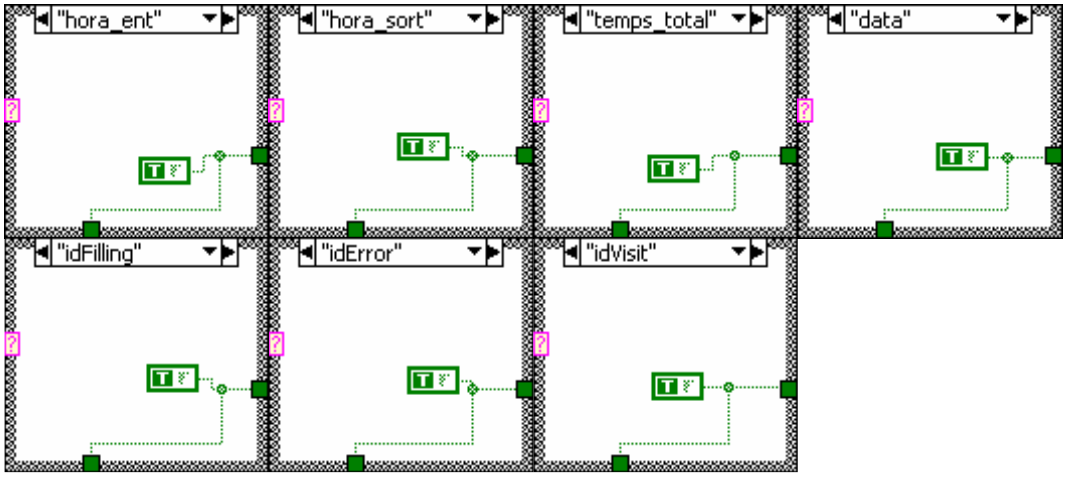


Figura 6.15-d: Condicionals del subprograma *sub-vi validaSelect.vi.*

sub-vi obtenirTaulesDelFrom.vi_____

Aquest sub-vi insertarà els noms que han de sortir en la taula resultant de la consulta, en un *array* per després poder tractar-lo millor al programa principal de consulta a la BD.

Arribarà un *string* amb els noms separats per comes (serà el contingut del *select*) i sortirà un *array* de *strings* amb un nom per cel·la.

Connector Pane

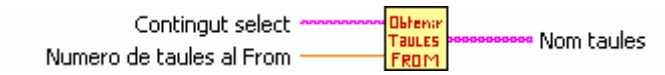


Figura 6.16-a: Connector Pane del subprograma *sub-vi obtenirTaulesDelFrom.vi.*

Front Panel

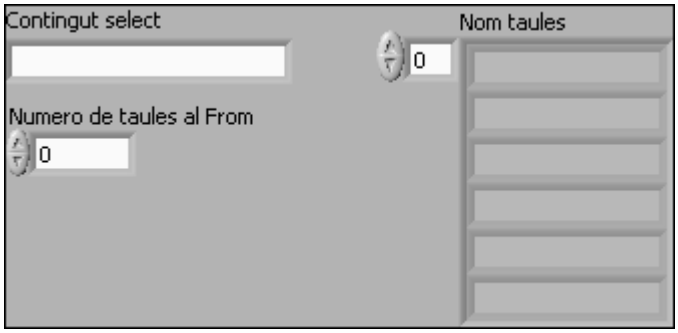


Figura 6.16-b: Front Panel del subprograma *sub-vi obtenirTaulesDelFrom.vi*

Block Diagram

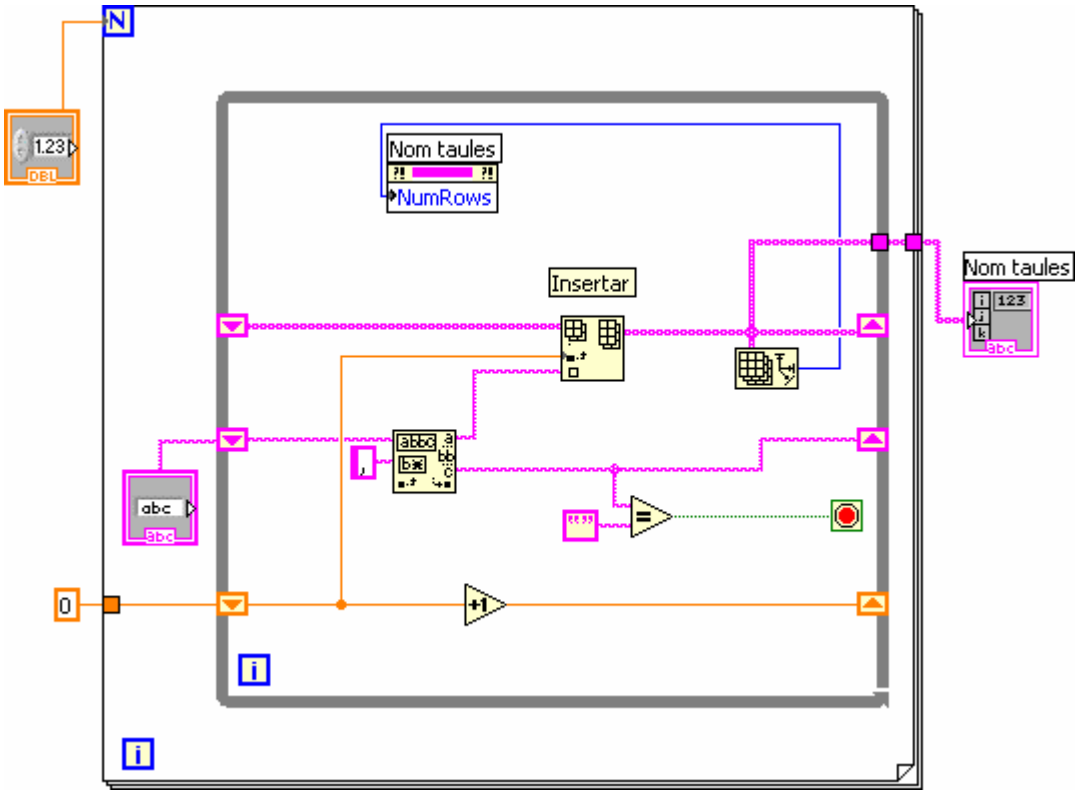


Figura 6.16-c: Block Diagram del subprograma *sub-vi obtenirTaulesDelFrom.vi.*

6.3.2. Consultar a la base de dades remotament utilitzant *webServer* de *LabVIEW*.

consulta_remota.vi

Aquest programa realitzarà consultes a una base de dades predeterminada, de forma remota utilitzant el *WebServer* de *LabVIEW*.

Aquest programa estarà basat en l'anterior de consultes a la BD, i més a més, utilitzarem el servidor web de *LabVIEW* per "transformar" el programa implementat en llenguatge *G*, en format de web dinàmica (*fitxer.htm*), el qual va refrescant constantment i es pot manipular el programa com si es tractés d'una execució local.

Per fer-ho utilitzarem dues eines fonamentals, el *Web Publishing tool* i el *Web Server (configuration)*.

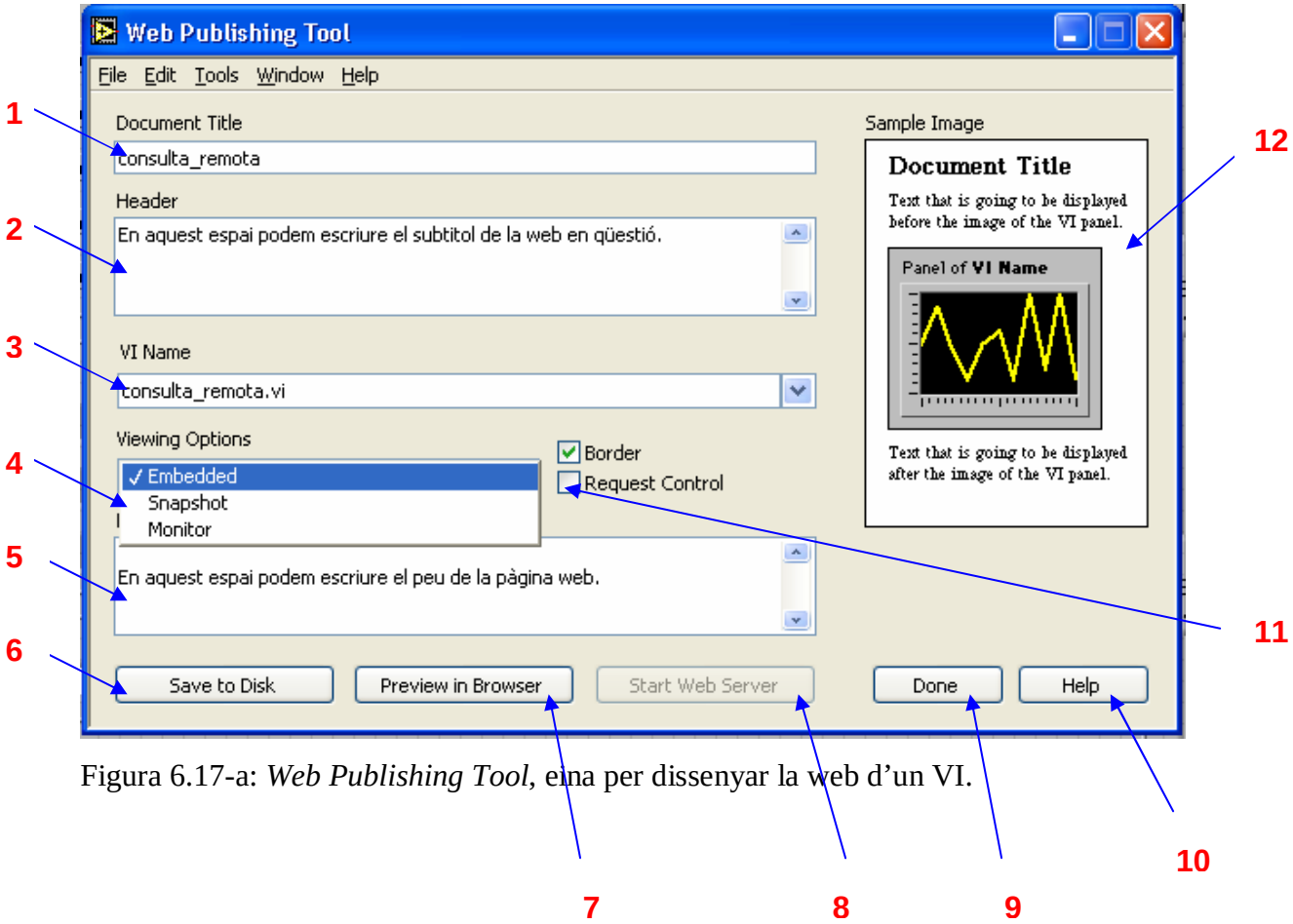


Figura 6.17-a: *Web Publishing Tool*, eina per dissenyar la web d'un VI.

6.3.2.1. *Web Publishing Tool* de *LabVIEW*.

És una eina que incorpora *LabVIEW* i que permet crear una pàgina web que estarà composta per un títol, un primer pàrraf, la imatge del panel frontal del programa en qüestió, o bé una simulació real del nostre panel frontal per poder controlar-lo i monitoritzar a distància l'aplicació utilitzant un navegador *http* i una connexió de xarxa.

Com es pot observar a la *figura 6.17-a*, la eina *Web Publishing Tool* té varies opcions per poder configurar una web a partir d'un *VI*. A continuació s'explicarà cada part de l'eina:

1. Títol de la web.
2. Paràgraf inicial on es pot escriure explicacions de funcionament de la web.
3. Seleccionar un *.vi* que s'hagi implementat en *LabVIEW* anteriorment.
4. Opcions en la que es mostrarà el pannell frontal del VI seleccionat: *Snapshot, Monitor o Embedded*. Les dues primeres són captures de pantalla (imatge estàtica) que poden anar refrescant de tant en tant per observar variacions de resultats. La opció més interessant per aquest treball és la de *Embedded*, on es pot controlar el pannell frontal del VI, amb la desavantatge que és la opció més pesada degut al enviament de dades constant.
5. Paràgraf final on es pot escriure el peu de la pàgina web.
6. Guardar la pàgina web al disc *(.htm)*.
7. Previsualitzar la pàgina web en un navegador.
8. Iniciar el servidor web.
9. Botó acceptar.
10. Botó ajuda.
11. Petició de control, s'ha d'activar per poder tenir el control exclusiu del VI, tant si es treballa des d'una màquina client com si es fa des del servidor.
12. Imatge de mostra.

6.3.2.2. *Web Server (configuration).*

És un servidor web que incorpora *LabVIEW* i que permet penjar un *VI* en format web per poder monitoritzar el panel frontal i per poder controlar-remotament. Per això es necessita un PC que faci de servidor i que tingui instal·lat *LabVIEW*, i tants PCs clients com es vulgui amb connexió de xarxa i tenir instal·lat el *RunTime* de *LabVIEW* (gratuit) i utilitzar un navegador *http* per visualitzar la web.

Com es pot observar a la *figura 6.17-b*, la eina *Web Server Configuration* té varies opcions per poder configurar el servidor web. A continuació s'explicarà cada part de l'eina:

1. Habilitar el servidor web de *LabVIEW*.
2. Directori arrel on es guardaran els fitxers *.htm* de les pàgines web.
3. Triar el port que s'utilitzarà pel protocol *HTTP* (port 80 per defecte).
4. Temps màxim de resposta per carregar la pàgina web.
5. Habilitar / deshabilitar el fitxer de registre de visites del servidor.

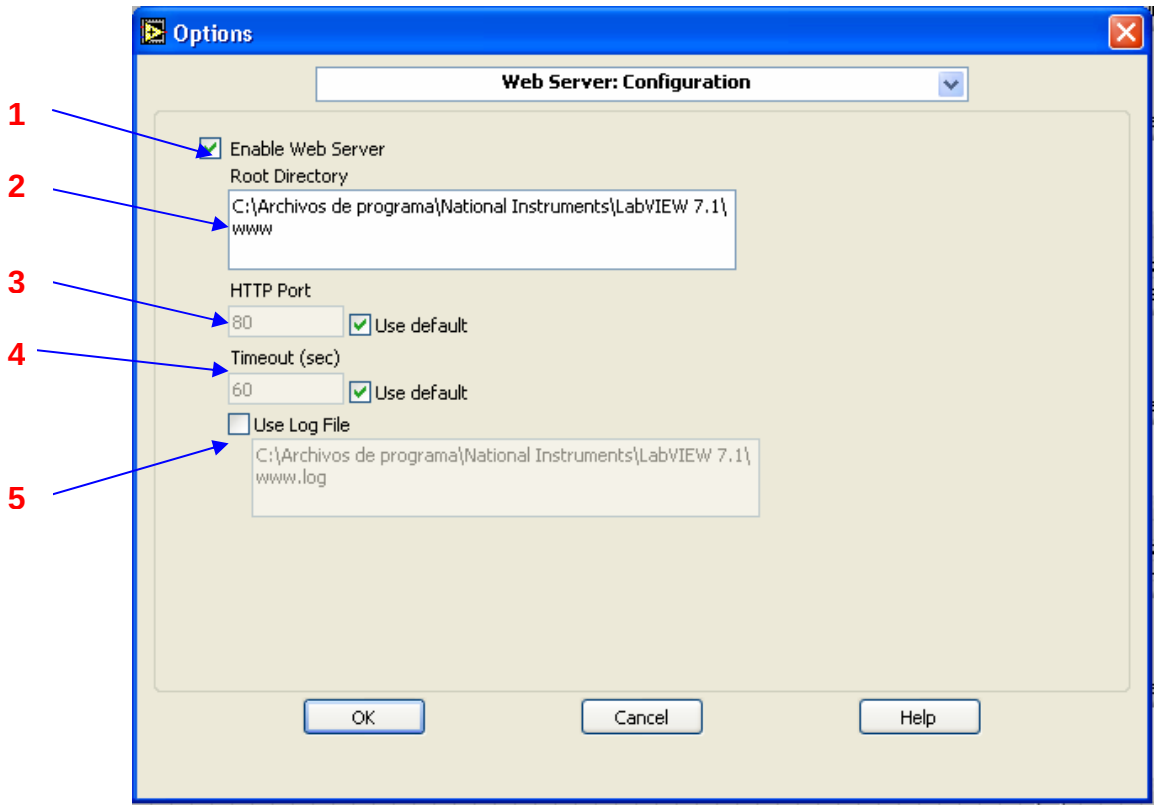


Figura 6.17-b: *Web Server (configuration)*, eina per configurar el servidor web.

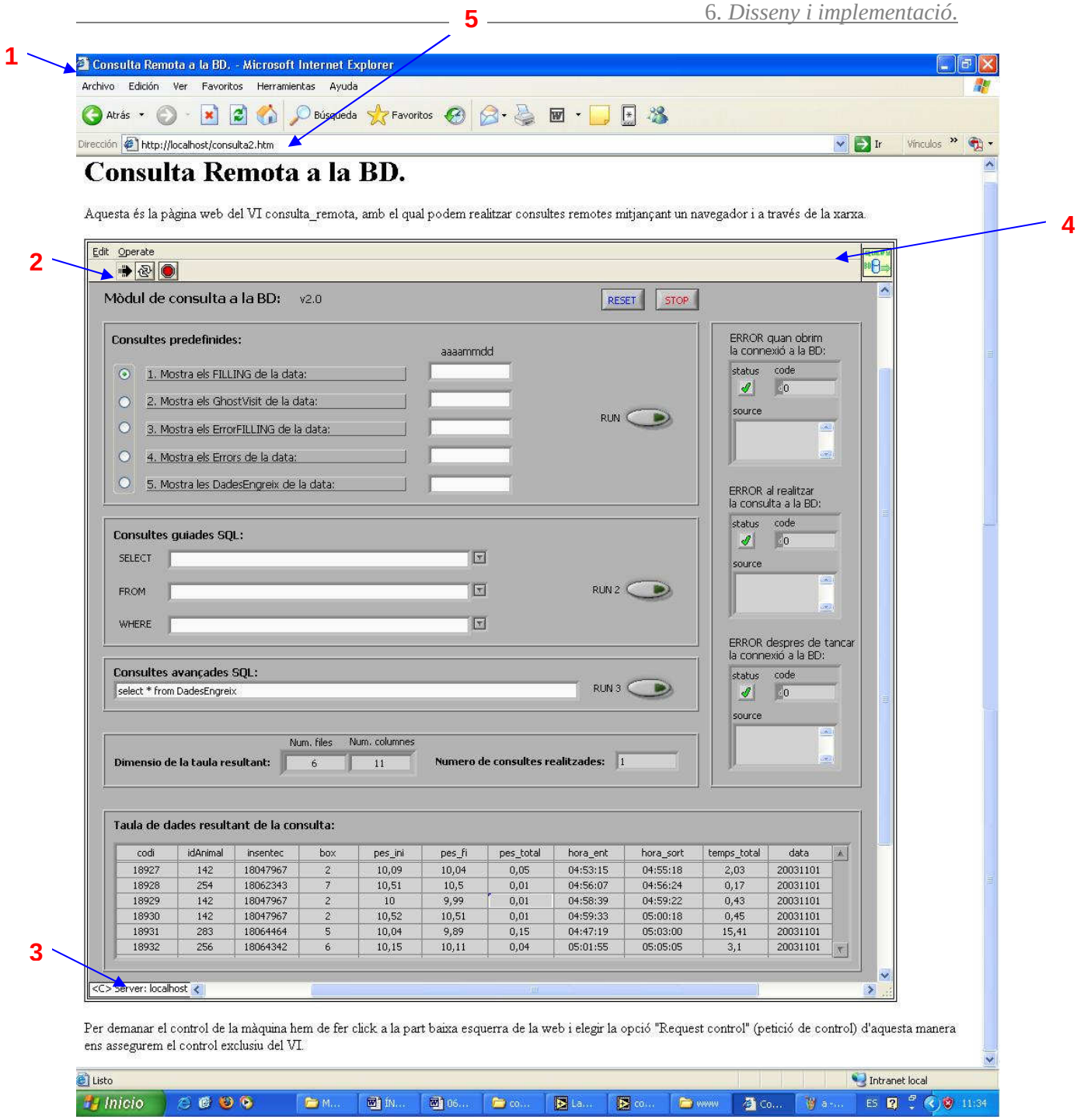


Figura 6.17-c: Pàgina web del programa consulta a la BD remotament.

A continuació s'explicarà les parts de la web del programa consulta remota a la BD:

1. Navegador web.
2. Executar el VI remotament.
3. Demanar la petició de control (*control request*).
4. Panell frontal remot.
5. Adreça de la web del tipus: http://adreça-IP/nom-fitxer.htm

6.4. Prototip de generació d'informes amb *Report Generation*.

En aquest apartat del treball s'explicarà la implementació del prototip de generació d'informes (*figura 6-E*), basat en el *VI* de consulta a al base de dades, afegint una funcionalitat de connexió amb un nou *VI*, en el qual triarem el tipus d'informe i les dades que volem analitzar.

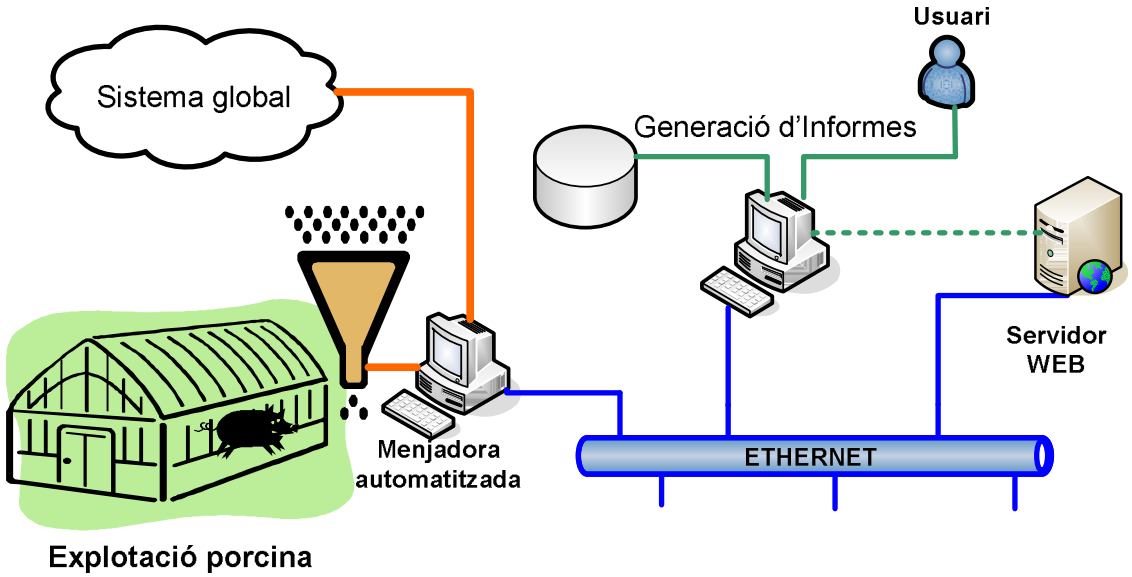


Figura 6-E: Diagrama de les eines de generació d'informes locals i remots via web.

6.4.1. Modificació del *VI* de consulta a la BD.

consulta2-1-4 genera infor 4-7.vi_____

Aquest programa és una modificació del *VI* de consulta a la BD que realitzarà consultes a una base de dades predeterminada, com el programa anterior de consultes, però a més a més afegeix una nova pestanya amb les consultes predeterminades per realitzar els posteriors informes.

El programa en qüestió consta de dos pestanyes: *consultes a la BD* i *generar informes.* La primera serveix per realitzar qualsevol tipus de consultes a la base de dades però no realitzarem informes amb elles. La segona, canvi, realitza uns tipus concrets de consultes, però els resultats d'aquestes consultes passaran a ser informes un cop s'enviï al programa de generació de informes.

Un cop feta la consulta es podrà fer click a un botó de “generar informe”, el qual enviarà les dades al programa *afegir dades a EXCEL1-2.vi* per poder allí triar el model de informe i generar-ho posteriorment.

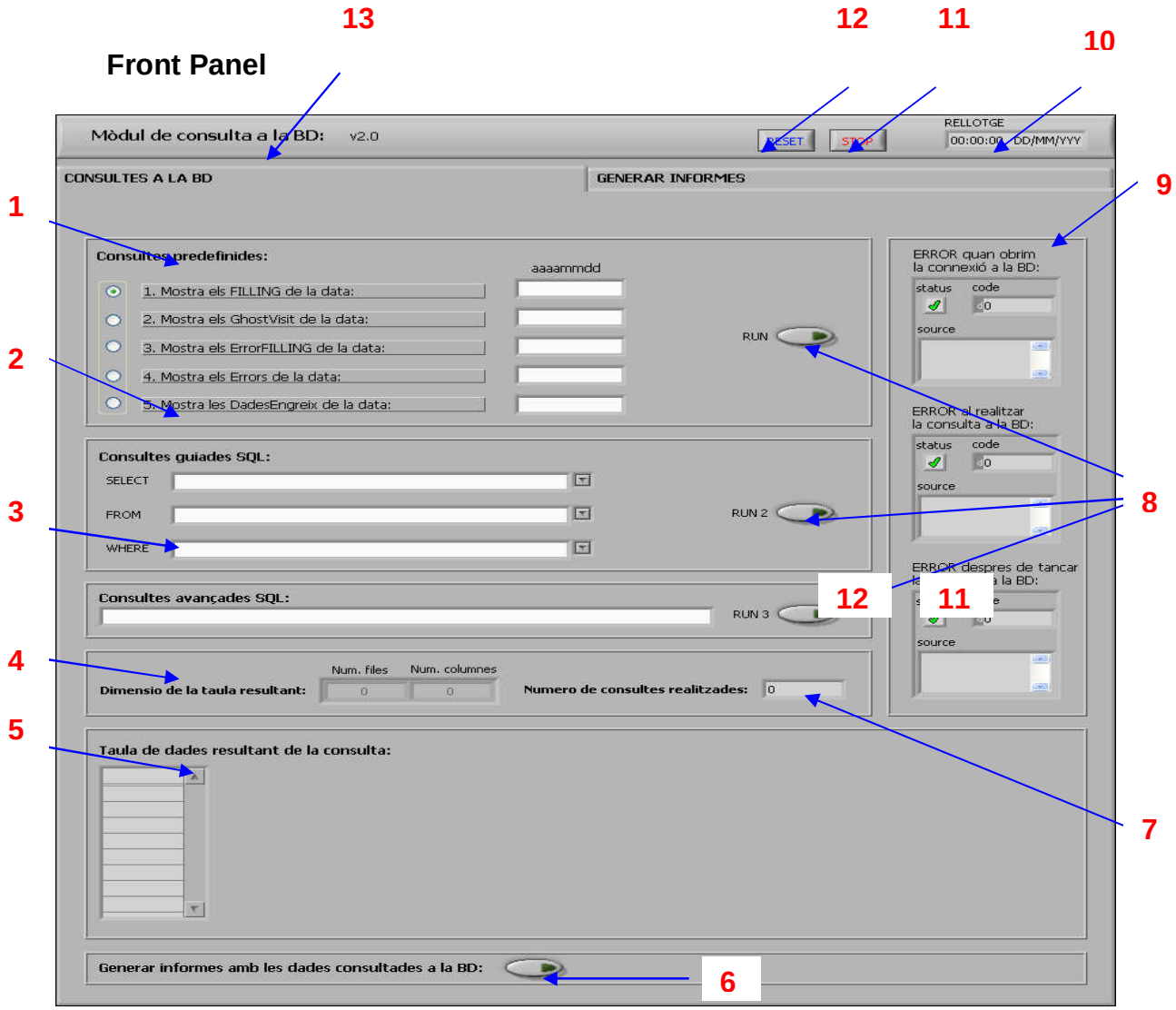


Figura 6.18-a: Front Panel de *consulta2-1-4 genera infor 4-7.vi* pestanya de consultes.

A continuació s’explicarà les diferents parts del panell frontal de la *figura 6.18-a*:

1. Consultes predefinides.
2. Consultes guiades on es triarà els parametres *SQL* mitjançant un desplegable.
3. Consultes avançades on s’escriurà la consulta sencera en *SQL*.
4. Dimensió de la taula resultant.
5. Taula resultant de la consulta.

6. Generar informes amb les dades resultants de la consulta.
7. Número de consultes realitzades.
8. Botons d'execució de les consultes.
9. Quadres de possibles errors de connexió, execució i desconnexió de la BD.
10. Data i hora del sistema.
11. Botó STOP.
12. Botó RESET.
13. Pestanya de consultes a la BD.

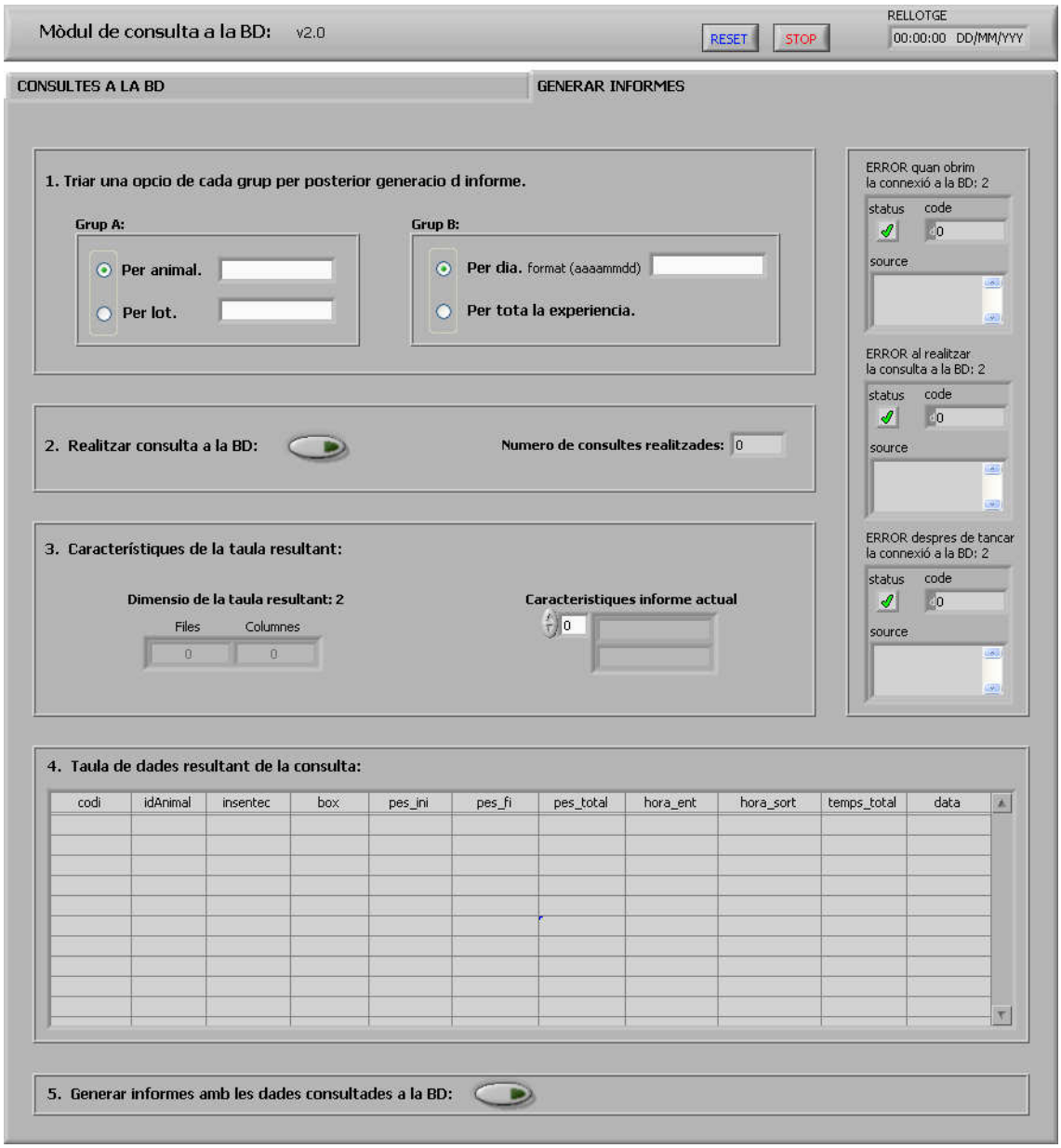


Figura 6.18-b: Front Panel de *consulta2-1-4 genera infor 4-7.vi* pestanya d'informes.

Block Diagram

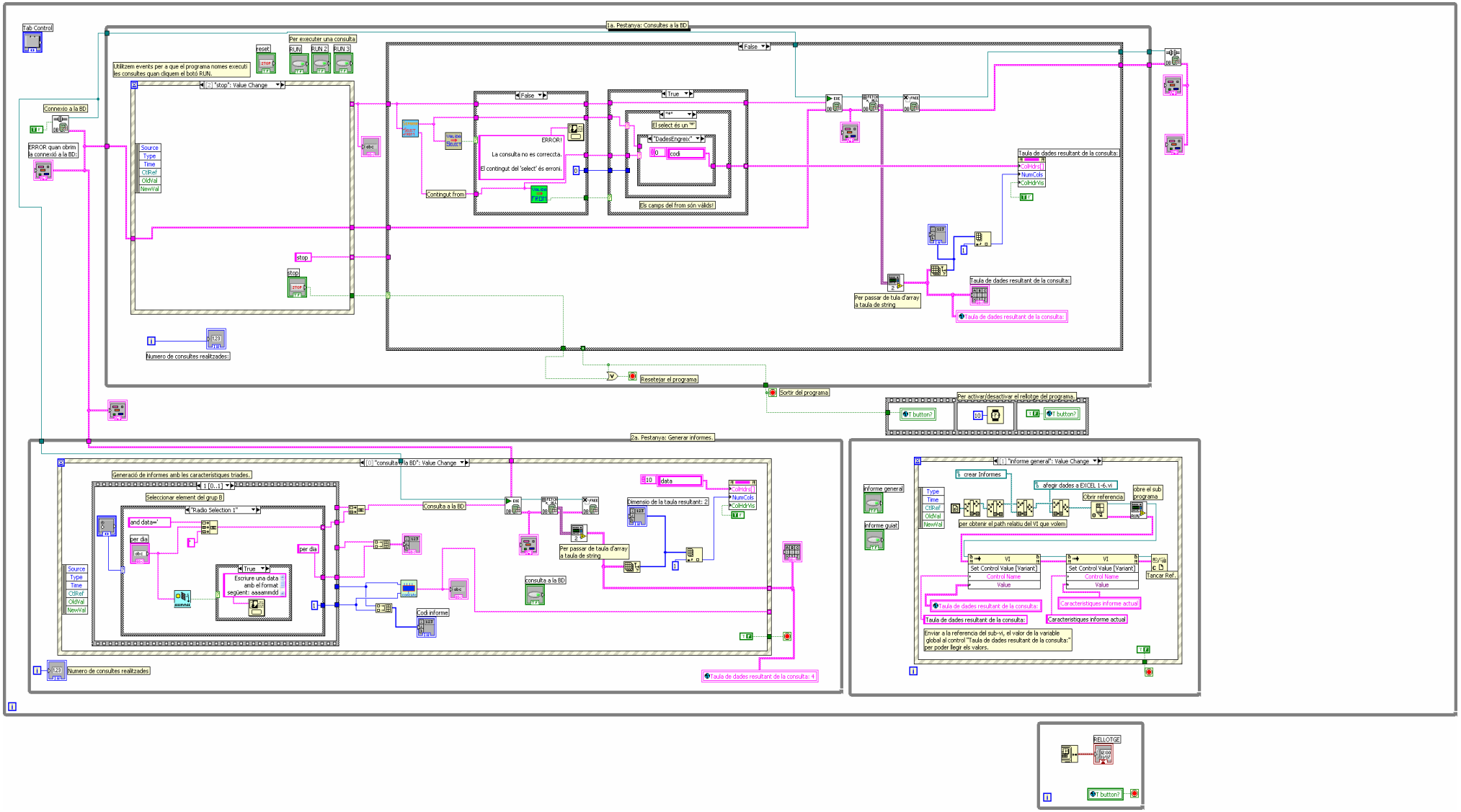


Figura 6.18-c: Block Diagram de *consulta2-1-4 genera infor 4-7.vi*.

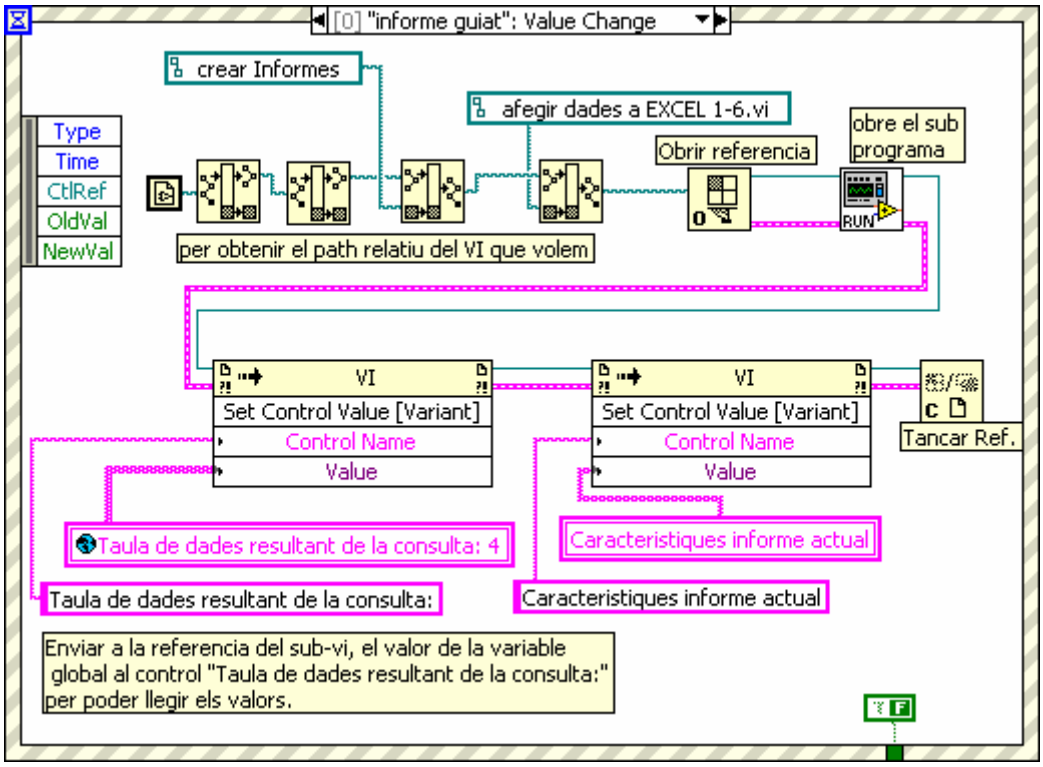


Figura 6.18-d: Codi del *event case* de *informe guiat* de *consulta2-1-4 genera infor 4-7.vi*

Llistat de subVIs del programa *guiat* de *consulta2-1-4 genera infor 4-7.vi.*

DB Tools Close Connection.vi
Connectar amb la BD.

DB Tools Execute Query.vi
Executar consulta a la BD.

DB Tools Fetch Recordset Data.vi
Extreure les dades obtingudes de una consulta en una *array* (matriu) 2D.

DB Tools Free Object.vi
Alliberar el objecte de la consulta destruint la referència associada.

sub-vi obtenirNomTaules.vi
Obtenir el nom de la taula de la BD on es vol realitzar la consulta.

sub-vi separarConsulta.vi
Separar el contingut del *select* i el contigut del *from* per comprovar que son camps vàlds en la nostra BD.

sub-vi validarFrom.vi
Validar que el contingut del *from* és vàlid per la nostra BD.

sub-vi validarSelect.vi
Validar que el contingut del *select* és vàlid per la nostra BD.

sub-vi obtenirTaulesDelFrom.vi
Insertar el nom de les taules resultants de la consulta SQL, per ficar-los en un

array (matriu) per poder després ficar els títols a la taula resultant.

DB Tools Open Connection.vi
Obrir la connexió amb la BD.

open fp and run.vi
Obrir un altre VI i executar-ho.

sub-vi concatenar codi informe.vi
Concatenar els paràmetres característics dels informes per després enviar-ho a un altre VI que generarà aquests informes.

Conn Execute.vi
Executar una consulta SQL y passa a fora una referència.

sub-vi canvi array a taula string.vi
Transformar una matriu (*array*) a una taula de *strings*.

global stop the program.vi
Variable global *booleana* per parar un VI.

sub-vi filtratgeData.vi
Validar el format de la data. Format correcte: aaaammdd.
(a=any, m=mes, d=dia).

globals.vi
Taula (*strings*) amb el resultat de la consulta realitzada per passar-la posteriorment a un altre VI que generarà l'informe.

sub-vi concatenar codi informe.vi_____

La funció d'aquest sub-instrument és concatenar els números que li entren per separats, i fer que surtint com a un *string* de caràcters.

Connector Pane

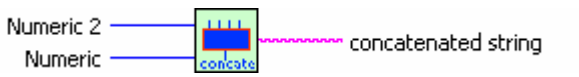


Figura 6.19-a: Connector Pane del *sub-vi concatenar codi informe.vi.*

Front Panel

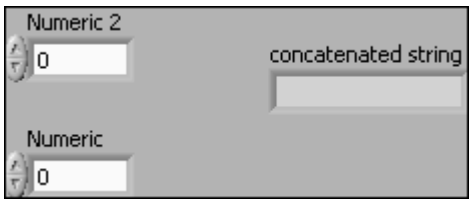


Figura 6.19-b: Front Panel del *sub-vi concatenar codi informe.vi.*

Block Diagram

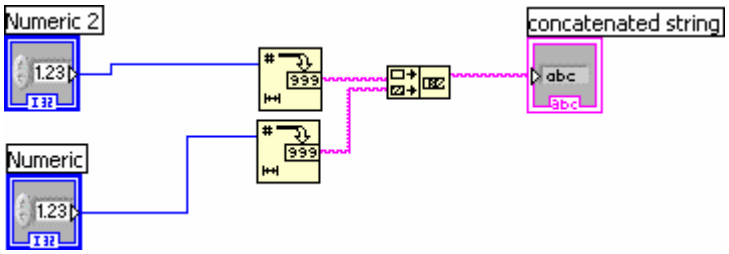


Figura 6.19-c: Block Diagram del *sub-vi concatenar codi informe.vi.*

sub-vi canvi array a taula string.vi

Connector Pane

Figura 6.20-a: Connector Pane del *sub-vi canvi array a taula string.vi.*

Front Panel

Figura 6.20-b: Front Panel del *sub-vi canvi array a taula string.vi.*

Block Diagram

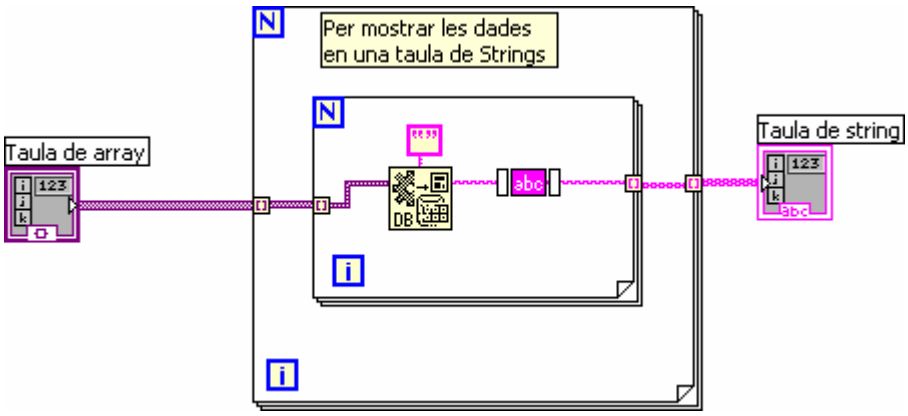


Figura 6.20-c: Block Diagram del *sub-vi canvi array a taula string.vi.*

Llista de sub-VIs utilitzats en el *sub-vi canvi array a taula string.vi.*

Trim Whitespace.vi
Deixa un espai en blanc.

sub-vi filtratgeData.vi

La funció d'aquest sub-vi és la de verificar que el format de la data que li entra (*string*) és correcte o no mitjançant un *booleà* de sortida.

Connector Pane

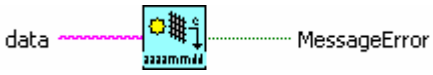


Figura 6.21-a: Connector Pane del *sub-vi filtratgeData.vi.*

Front Panel

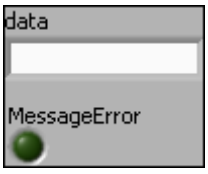


Figura 6.21-b: Front Panel del *sub-vi filtratgeData.vi.*

Block Diagram

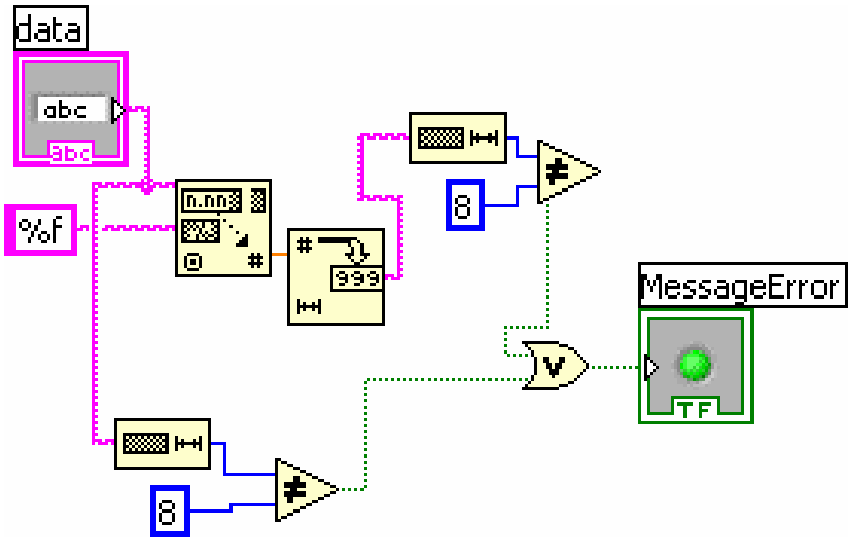


Figura 6.21-c: Block Diagram del *sub-vi filtratgeData.vi.*

6.4.2. Configuració i generació d'informes.

afegir dades a EXCEL 1-7.vi

Aquest programa és el complement de l'anterior, que realitza la consulta i l'enviarà a aquest, per realitzar la generació d'informes.

La funcionalitat d'aquest programa és la de configurar els paràmetres que es volen en l'informe que es generarà a partir d'aquest *VI* sobre el programa *MS EXCEL 2003.*

Connector Pane

Figura 6.22-a: Connector Pane del programa *afegir dades a EXCEL 1-7.vi.*

Front Panel

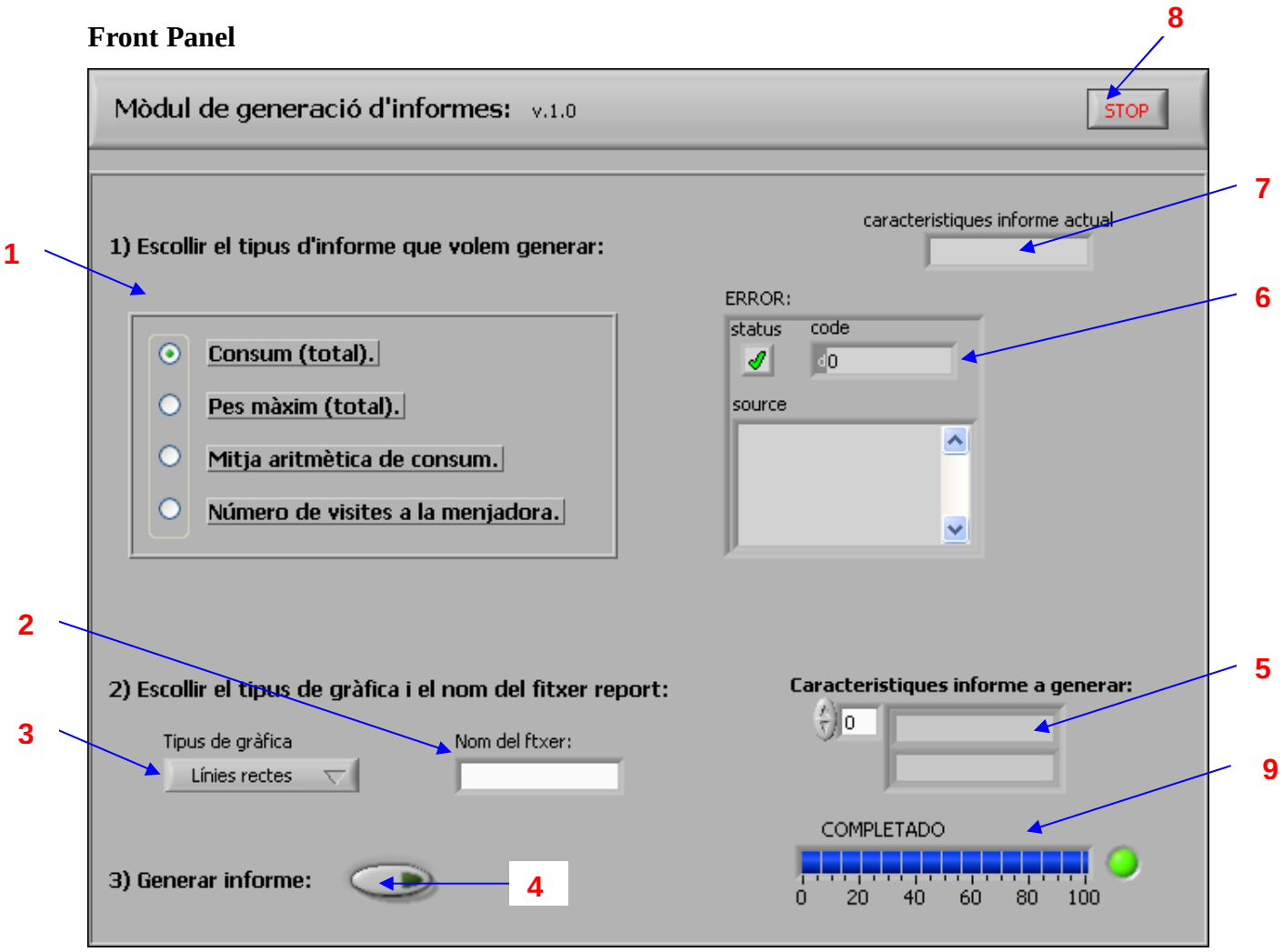


Figura 6.22-b: Front Panel del programa *afegir dades a EXCEL 1-7.vi.*

A continuació s'explicarà cada part del Front Panel:

1. Tipus d'informe que es vol generar a partir de la consulta que s'haurà fet anteriorment al programa de constulta a la BD.
2. Escriure el nom del fitxer de l'informe.
3. Escollir el tipus de gràfica de l'informe.
4. Botó generar informe.
5. Característiques de l'informe, en funció de la consulta realitzada.
6. Quadre de possibles errors.
7. Característiques informe actual.
8. Botó de STOP, per parar l'execució del programa.
9. Barra de procés (indica si el procés s'està carregant o està complet).

Block Diagram

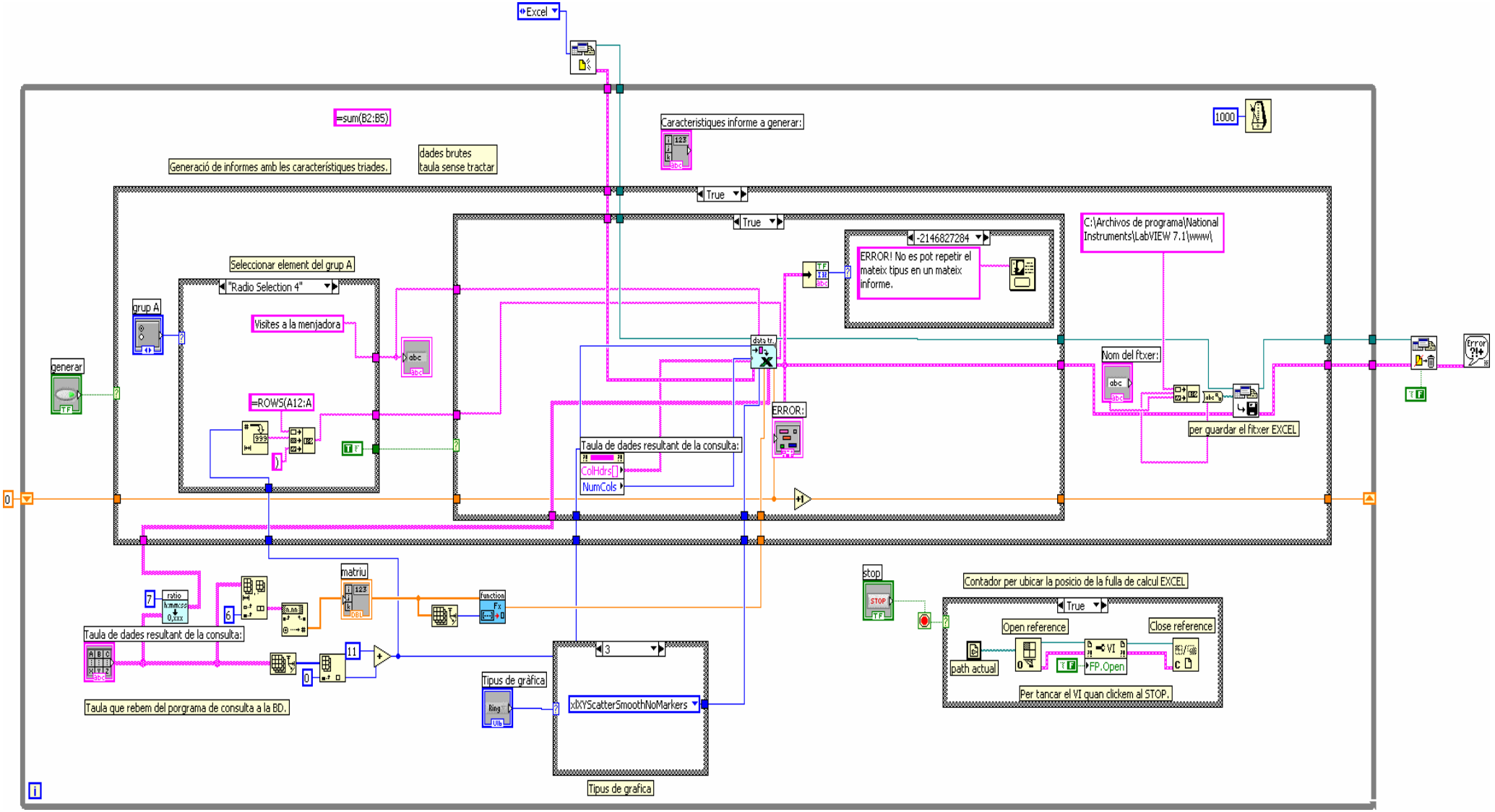


Figura 6.22-c: Block Diagram del programa *afegir dades a EXCEL 1-7.vi.*

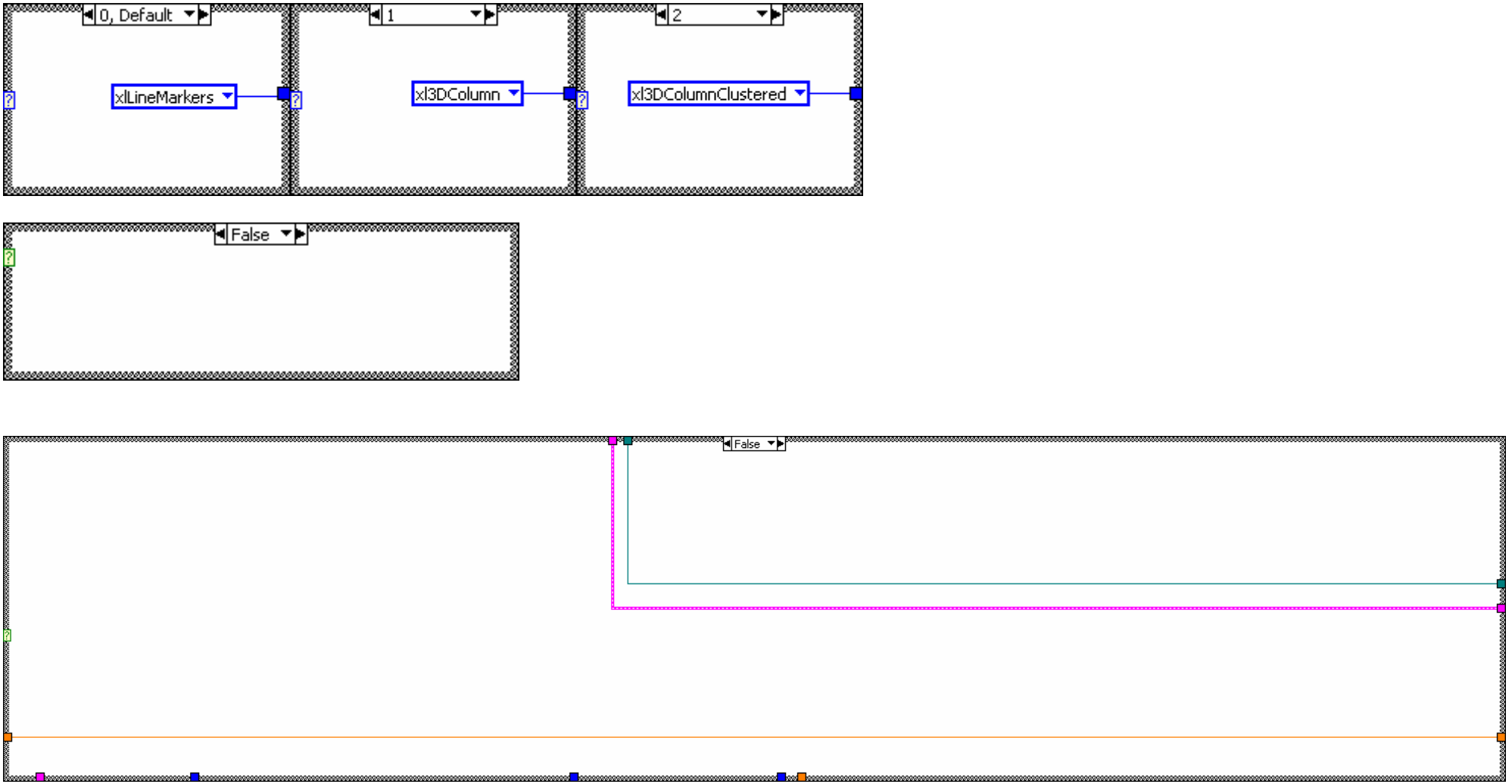


Figura 6.22-d: Diferents condicionals del programa *afegir dades a EXCEL 1-7.vi.*

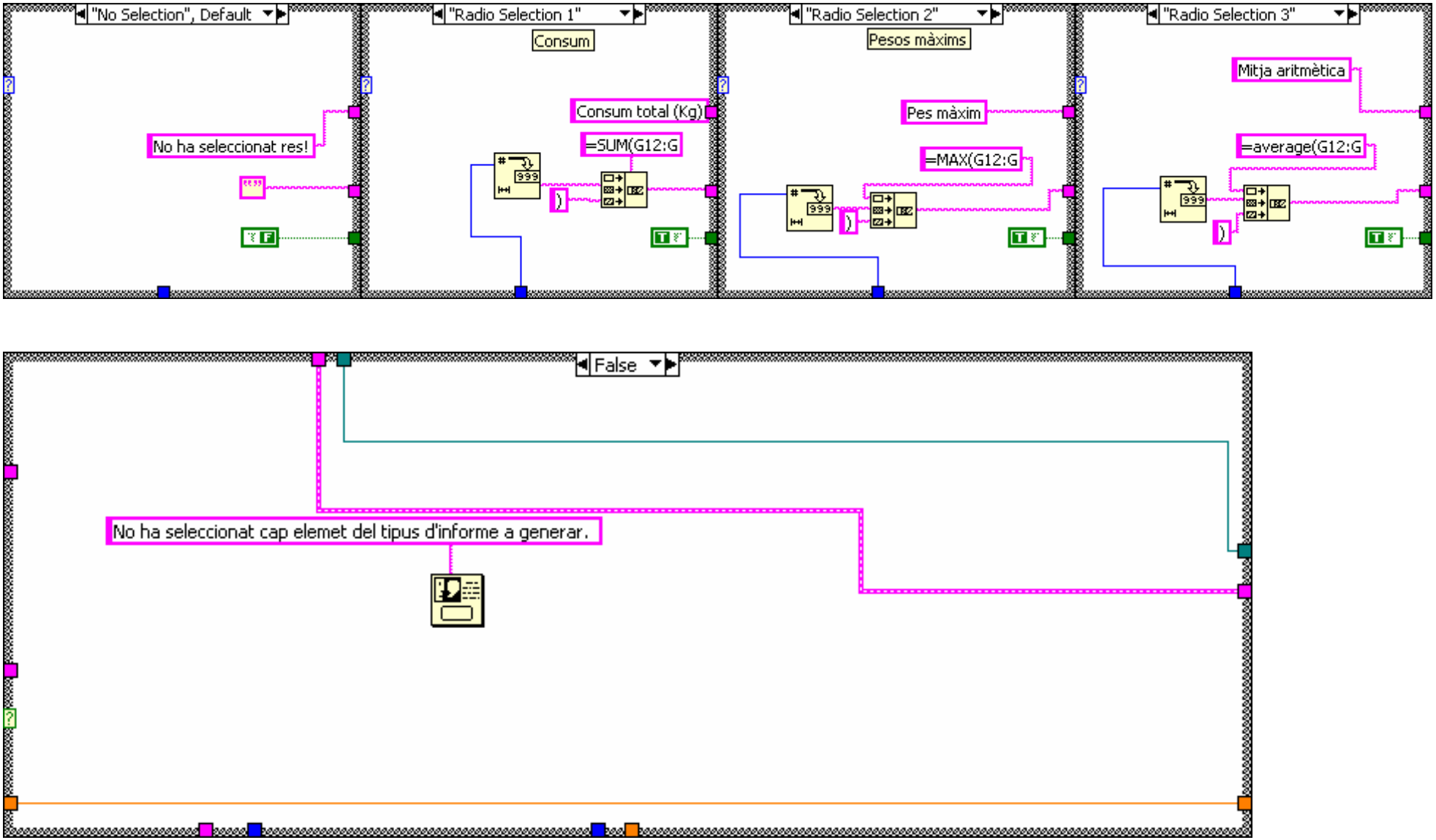


Figura 6.22-e: Diferents condicionals del programa *afegir dades a EXCEL 1-7.vi*

Llistat dels sub-VIs del programa *afegir dades a EXCEL 1-7.vi.*

sub-vi tractament dades 5.vi
Tractar les dades que han de sortir a l'informe generat. També te la funcionalitat de configurar tots els paràmetres de l'informe pròpiament.

New Report.vi
Crear un nou report, amb *Report Generation*.

sub-vi passar a segons 2.vi
Transformar una hora en format hh:mm:ss (h=hora, m=minuts, s=segons) a un ratio (numero de 0 a 1) que representen els segons totals, perquè a l'hora de generar la gràfica ho accepti.

sub-vi calcular mitja.vi
Calcular la mitja de un llistat de valors numerics.

General Error Handler.vi
Disparador d'ERRORS.

Dispose Report.vi
Tancar l'informe i alliberar la memòria utilitzada pel mateix.

Save Report to File.vi
Guardar un informe en un fitxer.

sub-vi tractament dades 6.vi_____

Aquest subVI és l'encarregat de tractar les dades que han de sortir a l'informe generat. També te la funcionalitat de configurar tots els paràmetres de l'informe pròpiament.

Connector Pane

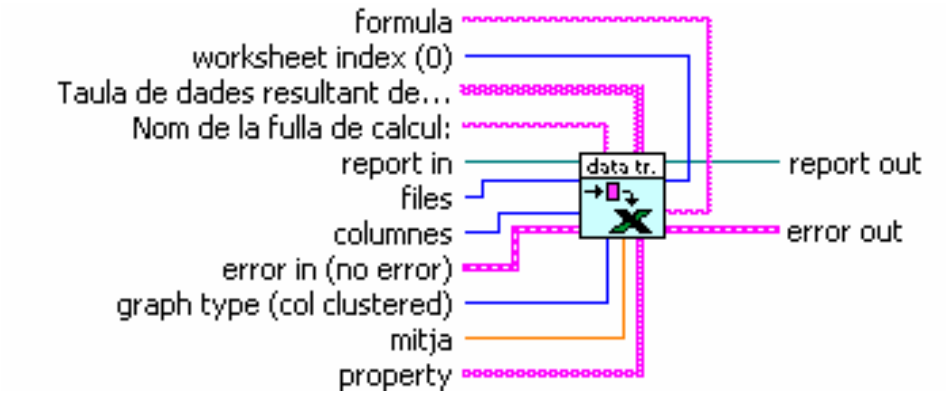


Figura 6.23-a: Connector Pane del subprograma *sub-vi tractament dades.vi.*

Front Panel

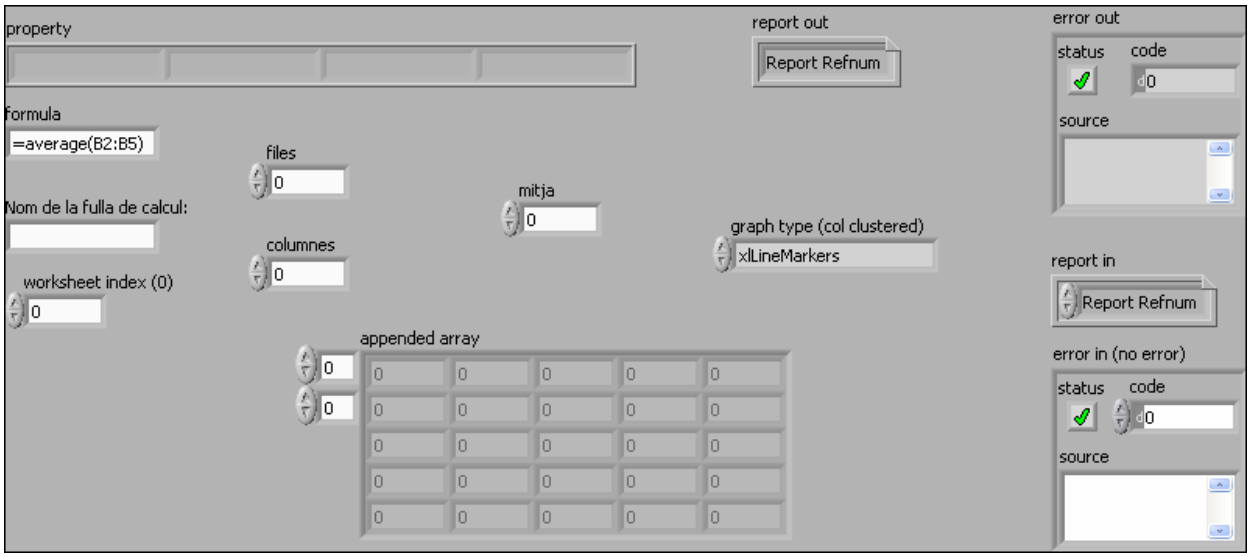
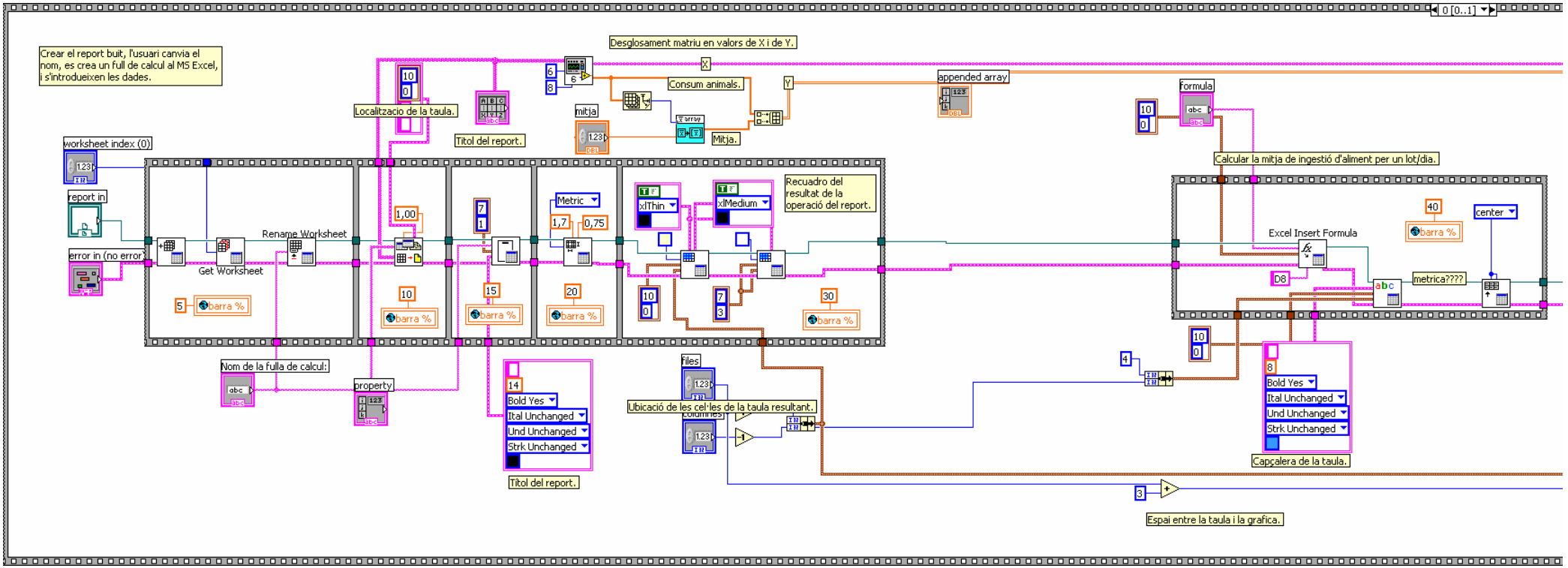


Figura 6.23-b: Front Paneldel subprograma *sub-vi tractament dades.vi.*

Block Diagram



*NOTA: Continuació a la pàgina següent.

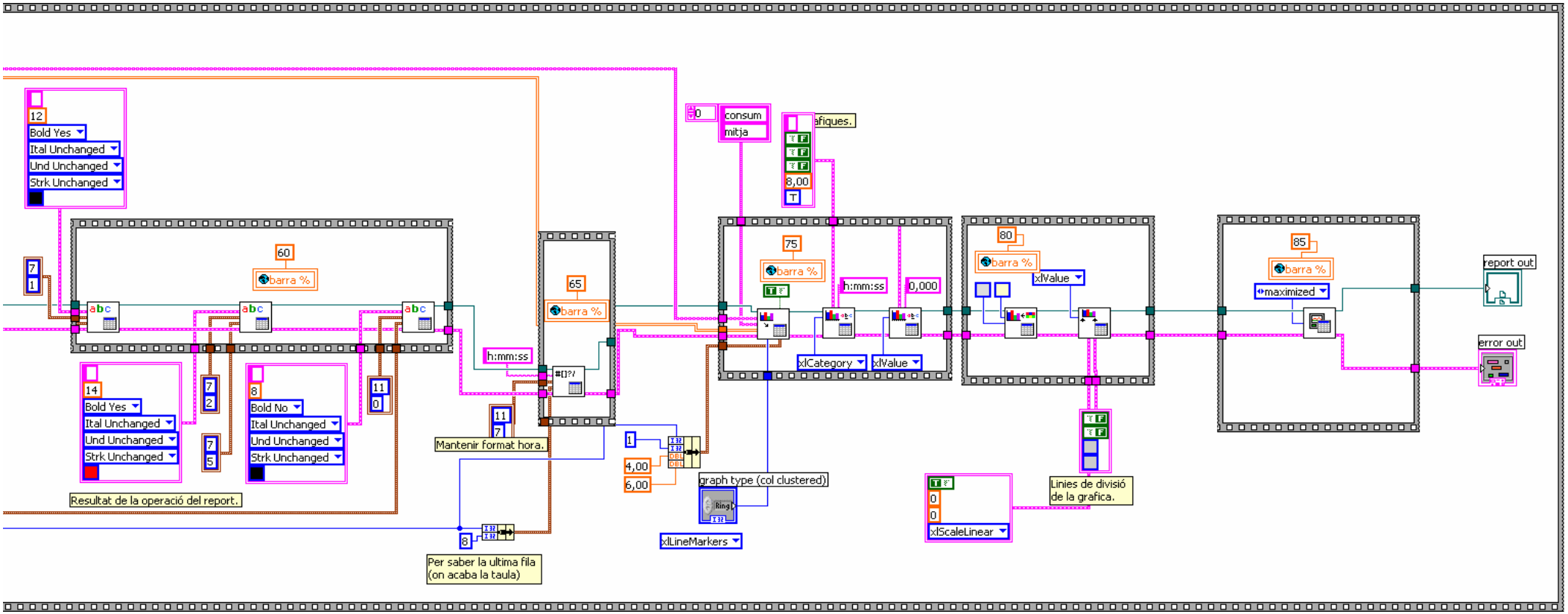


Figura 6.23-c: Block Diagram del subprograma *sub-vi tractament dades.*

Llistat de subVIs del *sub-vi tractament dades 5.vi.*

Append Table to Report.vi
Afegir taula al report actual.

Append Text Table to Report.vi
Afegir una taula al report actual.

Excel Get Worksheet.vi
Obtenir fulla de càlcul al *Ms Excel.*

Excel Rename Worksheet.vi
Renombrar una fulla de càlcul de *Ms Excel*.

Excel Insert Formula.vi
Insertar formula al *Ms Excel.*

Excel Add Worksheet.vi
Afegir fulla de càlcul.

Excel Set Cell Alignment.vi
Alinear cel·les de *Ms Excel.*

Excel Set Cell Dimension.vi
Dimensionar cel·les de *Ms Excel.*

Excel Easy Title.vi
Escriure títol de l'informe al *Ms Excel.*

Excel Set Cell Color and Border.vi
Configurar paràmetres de color i bordejats de les cel·les del *Ms Excel.*

Excel Set Cell Font.vi
Triar font de les lletres del informe de *Ms Excel.*

Excel Set Cell Format.vi
Triar format de cel·la al *Ms Excel.*

sub-vi eleccio columnes taula 2.vi
Elegir en quines columnes volem inserir dades al *Ms Excel.*

Excel Insert Graph.vi
Insertar una gràfica al *Ms Excel.*

Excel Set Graph Font.vi
Triar la font de les lletres de la gràfica.

Excel Set Graph Colors.vi
Triar els colors del paràmetres de la gràfica.

Excel Set Graph Scale.vi
Triar l'escala de la gràfica.

sub-vi creacio matriu de mitja.vi
Introduir els resultats de la mitja en una matriu 2D.

sub-vi paste png to excel.vi
Pegar una imatge *.png* en un informe de *MS Excel.*

sub-vi eleccio columnes taula 2.vi_____

Aquest subprograma té la funcionalitat de discriminar les columnes que volem d'una taula determinada. Entra una taula i el numero d'ordre de les dues columnes que volem i sortirà dues matrius, una de *strings* amb els headers (títols de les columnes triades) i l'altra, de tipus numèrica, amb els valors d'aquestes columnes.

Connector Pane

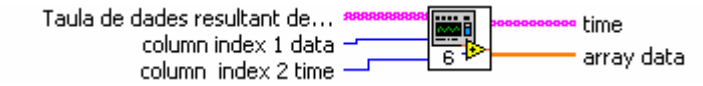


Figura 6.24-a: Connector Pane del *sub-vi eleccio columnes taula 2.vi.*

Front Panel

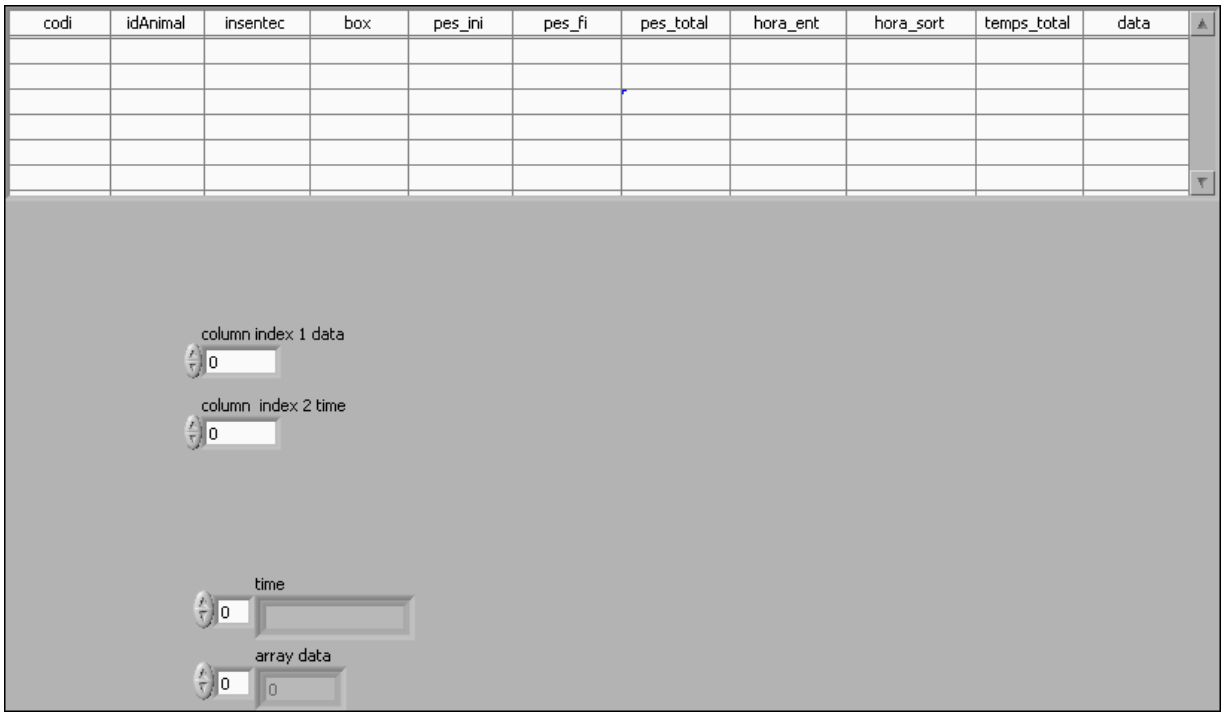


Figura 6.24-b: Front Panel del *sub-vi eleccio columnes taula 2.vi.*

Block Diagram

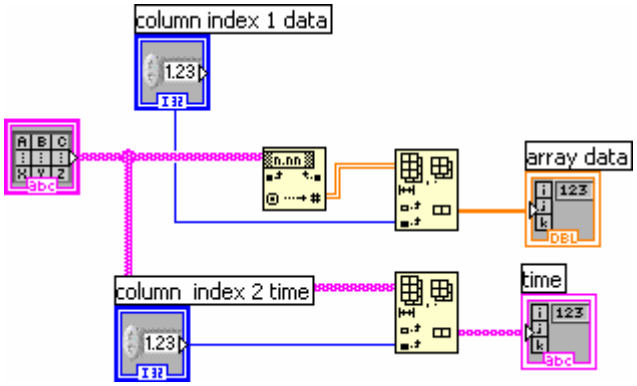


Figura 6.24-c: Block Diagram del *sub-vi eleccio columnes taula 2.vi.*

sub-vi creacio matriu de mitja.vi

Connector Pane

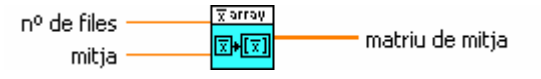


Figura 6.25-a: Connector Pane del *sub-vi creacio matriu de mitja.vi.*

Front Panel

Figura 6.25-b: Front Panel del *sub-vi creacio matriu de mitja.vi.*

Block Diagram

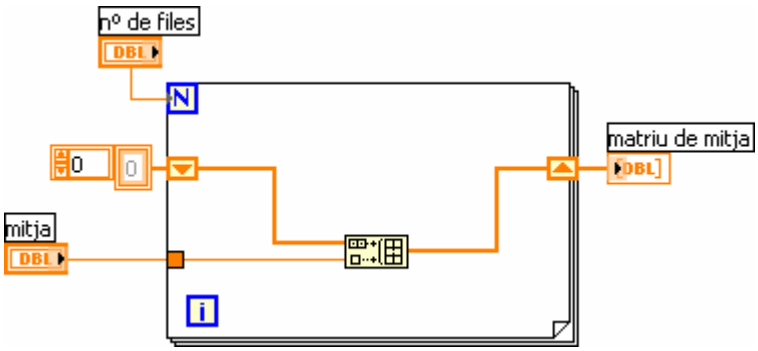


Figura 6.25-c: Block Diagram del *sub-vi creacio matriu de mitja.vi.*

sub-vi paste png to excel.vi_____

Connector Pane

Figura 6.26-a: Connector Pane del *sub-vi paste png to excel.vi.*

Front Panel

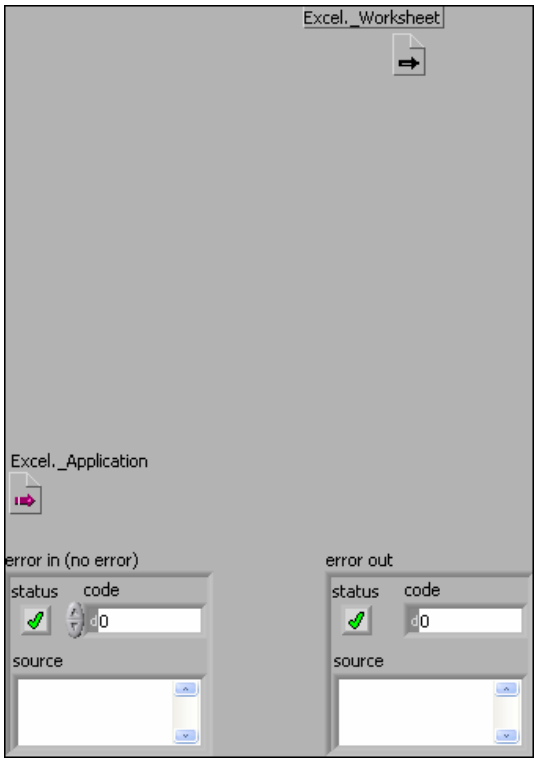


Figura 6.26-b: Block Diagram del *sub-vi paste png to excel.vi.*

Block Diagram

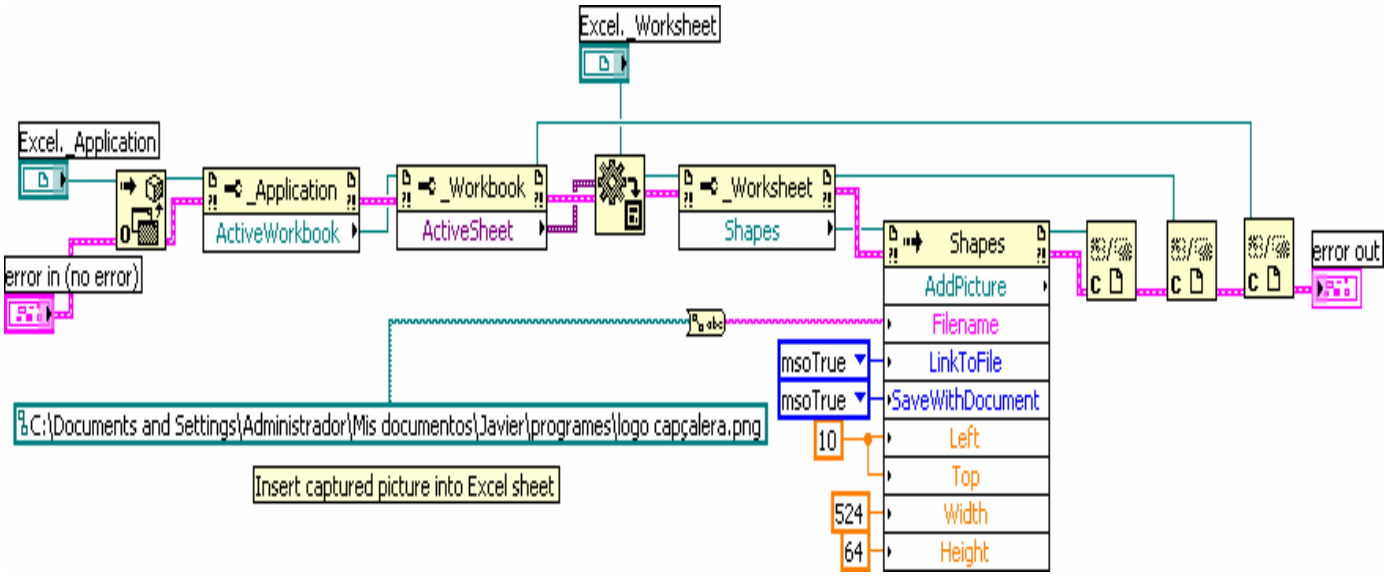


Figura 6.26-c: Block Diagram del *sub-vi paste png to excel.vi.*

sub-vi passar a segons 2.vi

Connector Pane

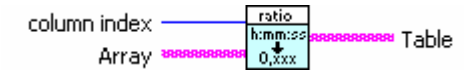


Figura 6.27-a: Connector Pane del *sub-vi passar a segons 2.vi.*

Front Panel

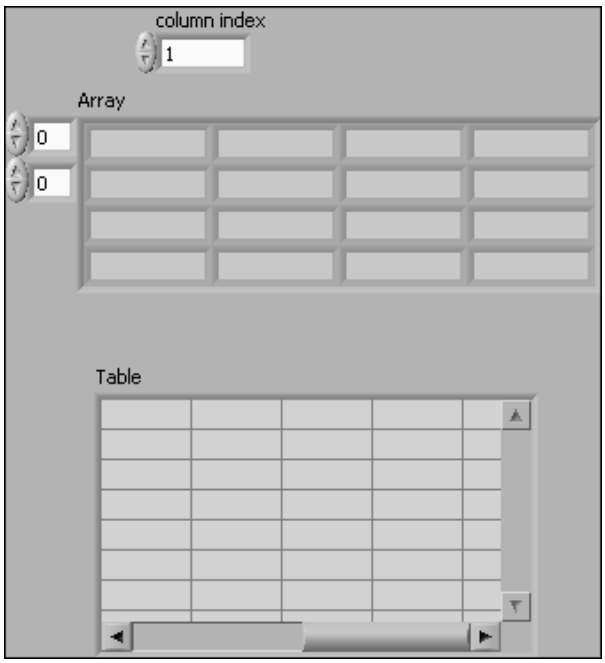


Figura 6.27-b: Block Diagram del *sub-vi passar a segons 2.vi.*

Block Diagram

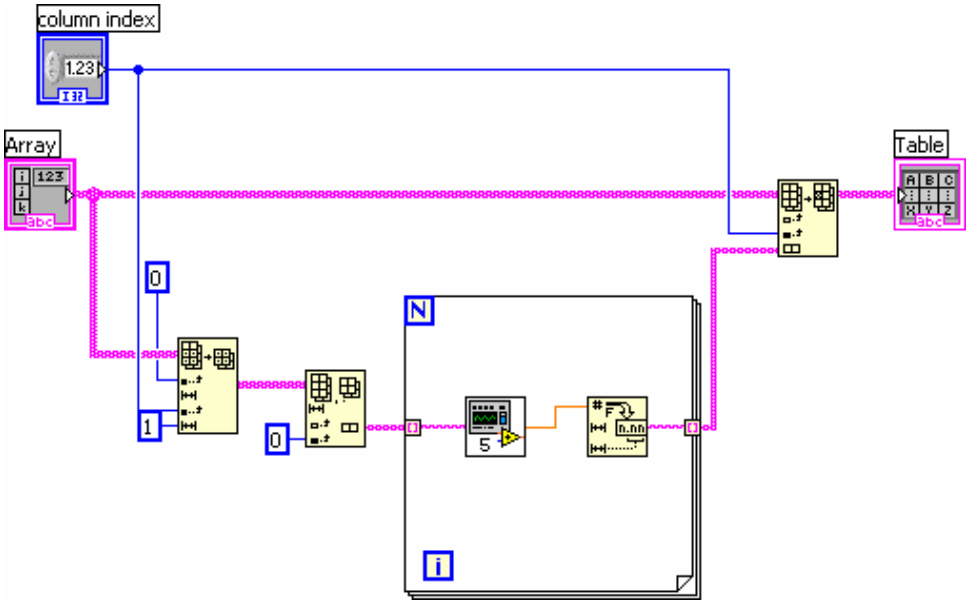


Figura 6.27-c: Block Diagram del *sub-vi passar a segons 2.vi.*

Llista de subVIs utilitzats en el *sub-vi passar a segons 2.vi.*

sub-vi ratio hora-numero.vi

Transformar el número totals de segons d'una hora determinada en un ratio (numero entre 0 i 1) per poder així interpretar-ho la gràfica del report en qüestió.

sub-vi ratio hora-numero.vi_____

Connector Pane

Figura 6.28-a: Connector Pane del *sub-vi ratio hora-numero.vi.*

Front Panel

Figura 6.28-b: Front Panel del *sub-vi ratio hora-numero.vi.*

Block Diagram

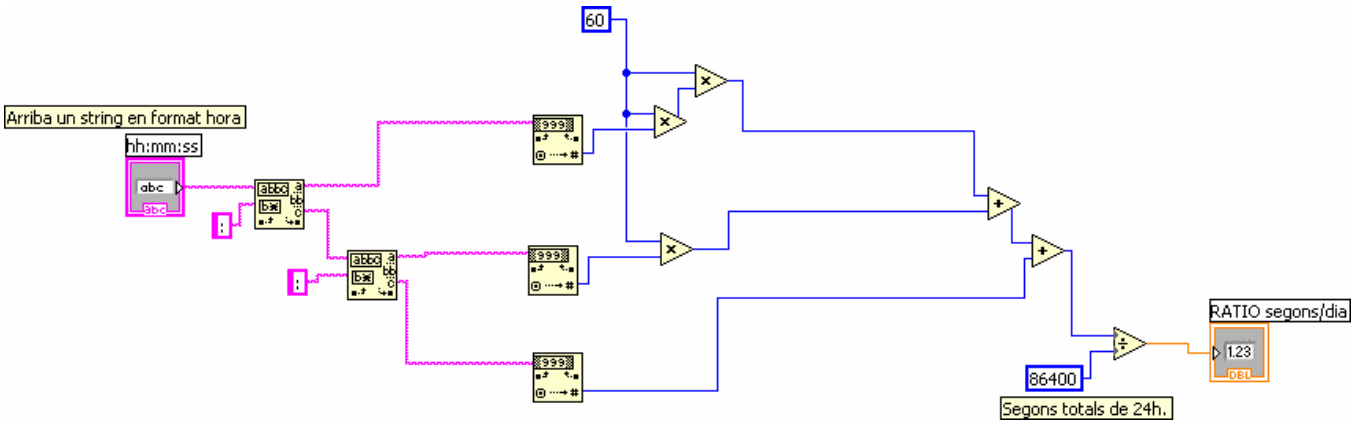


Figura 6.28-c: Block Diagram del *sub-vi ratio hora-numero.vi.*

sub-vi calcular mitja.vi_____

Connector Pane

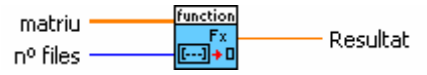


Figura 6.29-a: Connector Pane del *sub-vi calcular mitja.vi.*

Front Panel

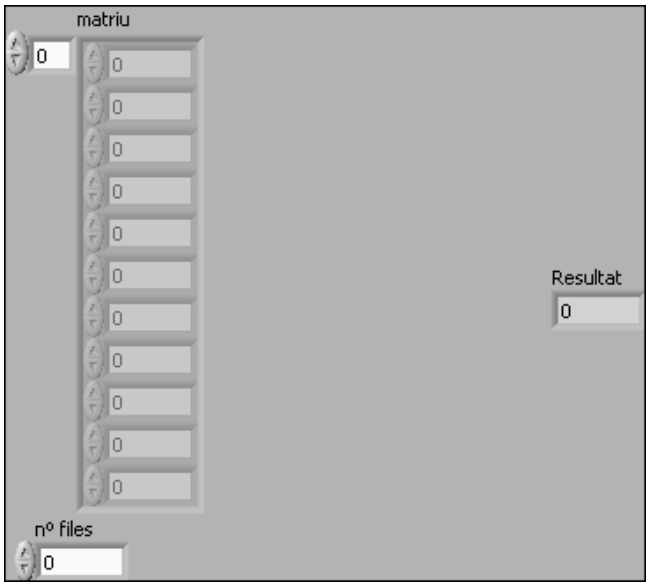


Figura 6.29-b: Front Panel del *sub-vi calcular mitja.vi.*

Block Diagram

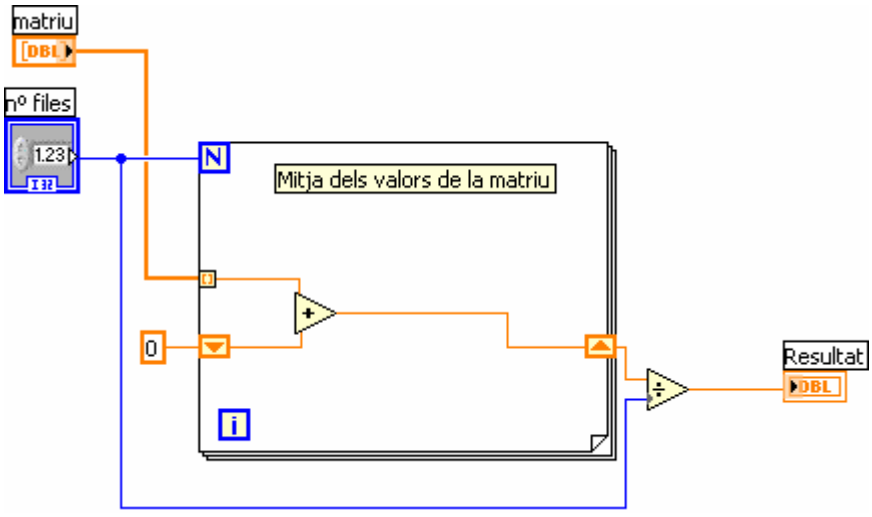


Figura 6.29-c: Block Diagram del *sub-vi calcular mitja.vi.*

7 Proves i avaluació

Capítol 7

Proves i avaluació.

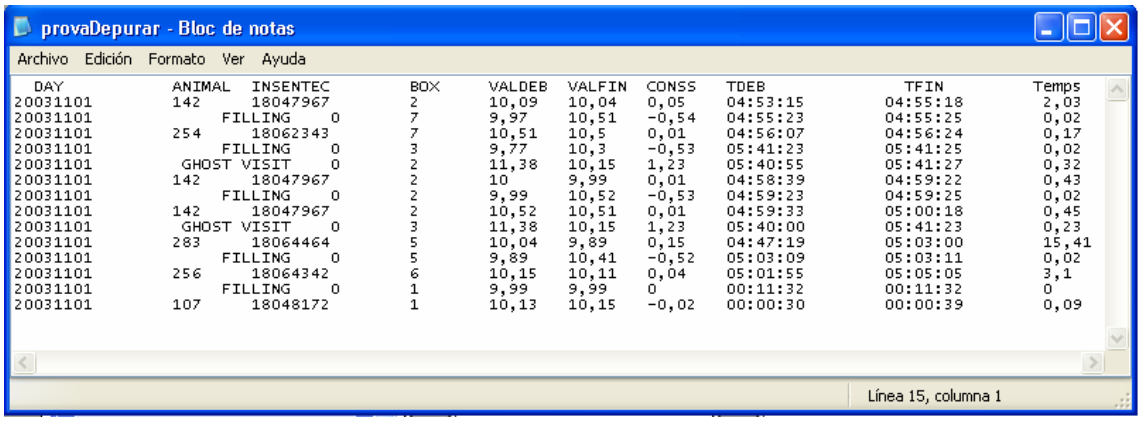
En aquest apartat s'exposarà l'avaluació dels programes dissenyats i implementats en aquests treball, així com les proves realitzades per verificar que el codi és fiable i compleix amb els objectius del treball.

7.1. Eina d'inserció i depuració de les dades a la BD.

Les proves realitzades en aquest punt del treball són les següents:

- Provar la integritat i l'eficiència del codi mitjançant proves d'inserció i depuració de dades reals obtingudes de la menjadora (d'altres experiències).

- Comprovar que la BD respon correctament emmagatzemant les dades rebudes del programa.



DAY	ANIMAL	INSENTEC	BOX	VALDEB	VALFIN	CONSS	TDEB	TFIN	Temps
20031101	142	18047967	2	10,09	10,04	0,05	04:53:15	04:55:18	2,03
20031101	FILLING	0	7	9,97	10,51	-0,54	04:55:23	04:55:25	0,02
20031101	254	18062343	7	10,51	10,5	0,01	04:56:07	04:56:24	0,17
20031101	FILLING	0	3	9,77	10,3	-0,53	05:41:23	05:41:25	0,02
20031101	GHOST VISIT	0	2	11,38	10,15	1,23	05:40:55	05:41:27	0,32
20031101	142	18047967	2	10	9,99	0,01	04:58:39	04:59:22	0,43
20031101	FILLING	0	2	9,99	10,52	-0,53	04:59:23	04:59:25	0,02
20031101	142	18047967	2	10,52	10,51	0,01	04:59:33	05:00:18	0,45
20031101	GHOST VISIT	0	3	11,38	10,15	1,23	05:40:00	05:41:23	0,23
20031101	283	18064464	5	10,04	9,89	0,15	04:47:19	05:03:00	15,41
20031101	FILLING	0	5	9,89	10,41	-0,52	05:03:09	05:03:11	0,02
20031101	256	18064342	6	10,15	10,11	0,04	05:01:55	05:05:05	3,1
20031101	FILLING	0	1	9,99	9,99	0	00:11:32	00:11:32	0
20031101	107	18048172	1	10,13	10,15	-0,02	00:00:30	00:00:39	0,09

Figura 7.1: Fitxer *provaDepurar.txt* (1 KB) amb 14 línies de dades.

La primera línia dels fitxers de dades s'ha de depreciar perquè són les capçaleres de les columnes i no són dades pròpiament.

provaGeneral - Bloc de notas

Archivo Edición Formato Ver Ayuda

DAY	ANIMAL	INSENTEC	BOX	VALDEB	VALFIN	CONSS	TDEB	TFIN	Temps
20031101	107	18048172	1	10,16	10,13	0,03	23:59:55	00:00:28	0,33
20031101	107	18048172	1	10,13	10,15	-0,02	00:00:30	00:00:39	0,09
20031101	107	18048172	1	10,15	10,13	0,02	00:00:39	00:00:46	0,07
20031101	125	18056567	3	10,41	10,28	0,13	23:53:36	00:00:46	7,1
20031101	107	18048172	1	10,13	10,12	0,01	00:00:48	00:01:25	0,37
20031101	107	18048172	1	10,12	10,09	0,03	00:01:28	00:01:57	0,29
20031101	107	18048172	1	10,1	10,1	0	00:01:58	00:02:00	0,02
20031101	107	18048172	1	10,1	10,11	-0,01	00:02:00	00:02:02	0,02
20031101	107	18048172	1	10,11	10,1	0,01	00:02:03	00:02:16	0,13
20031101	107	18048172	1	10,1	10,12	-0,02	00:02:17	00:02:27	0,1
20031101	107	18048172	1	10,12	10,1	0,02	00:02:28	00:02:32	0,04
20031101	107	18048172	1	10,1	10,09	0,01	00:02:33	00:02:41	0,08
20031101	283	18064464	5	10,52	10,47	0,05	23:59:47	00:02:46	2,59
20031101	107	18048172	1	10,09	10,09	0	00:02:42	00:02:56	0,14
20031101	146	18047882	2	10,1	10,09	0,01	00:02:37	00:02:56	0,19
20031101	167	18062230	7	10,05	10	0,05	23:57:37	00:03:01	5,24
20031101	107	18048172	1	10,1	10,1	0	00:02:59	00:03:04	0,05
20031101	FILLING	0	7	10	10,52	-0,52	00:03:11	00:03:13	0,02
20031101	129	18041735	4	9,83	9,61	0,22	23:54:58	00:03:14	8,16
20031101	FILLING	0	4	9,61	10,14	-0,53	00:03:25	00:03:27	0,02
20031101	107	18048172	1	10,12	10,09	0,03	00:03:08	00:03:28	0,2
20031101	129	18041735	4	10,14	10,14	0	00:03:36	00:03:45	0,09
20031101	107	18048172	1	10,09	10,09	0	00:03:47	00:03:49	0,02
20031101	107	18048172	1	10,09	10,09	0	00:03:49	00:03:51	0,02
20031101	147	18048225	3	10,28	10,21	0,07	00:00:46	00:03:54	3,08
20031101	255	18063151	5	10,47	10,46	0,01	00:03:47	00:04:12	0,25
20031101	114	18041545	1	10,09	10,09	0	00:04:04	00:04:15	0,11
20031101	256	18064342	6	10,06	10,05	0,01	00:04:41	00:05:49	1,08
20031101	167	18062230	7	10,52	10,51	0,01	00:04:15	00:05:50	1,35
20031101	255	18063151	5	10,46	10,44	0,02	00:04:45	00:06:00	1,15

Línea 3777, columna 1

Figura 7.2: Fitxer *provaGeneral.txt* (236 KB) amb 3776 línies de dades.

A la *figura 7.2* podem comprovar un fitxer de dades corresponent a un dia sencer d'una experiència en una granja pilot, el qual conté un nombre considerable de línies de dades les quals s'han d'emmagatzemar per poder gestionar-les correctament. El nombre real de línies de dades és 3776, una menys que el que indica el peu de la captura del *fitxer.txt*, ja que la primera no es pot contar perquè són les capçaleres de cada columna.

A la *figura 7.3* es pot comprovar que les dades han estat inserides correctament a la BD un cop han estat captades i filtrades pels programes corresponents.

El resultat global de l'avaluació és satisfactori, tant el programa com les bases de dades han funcionat correcta i eficientment.

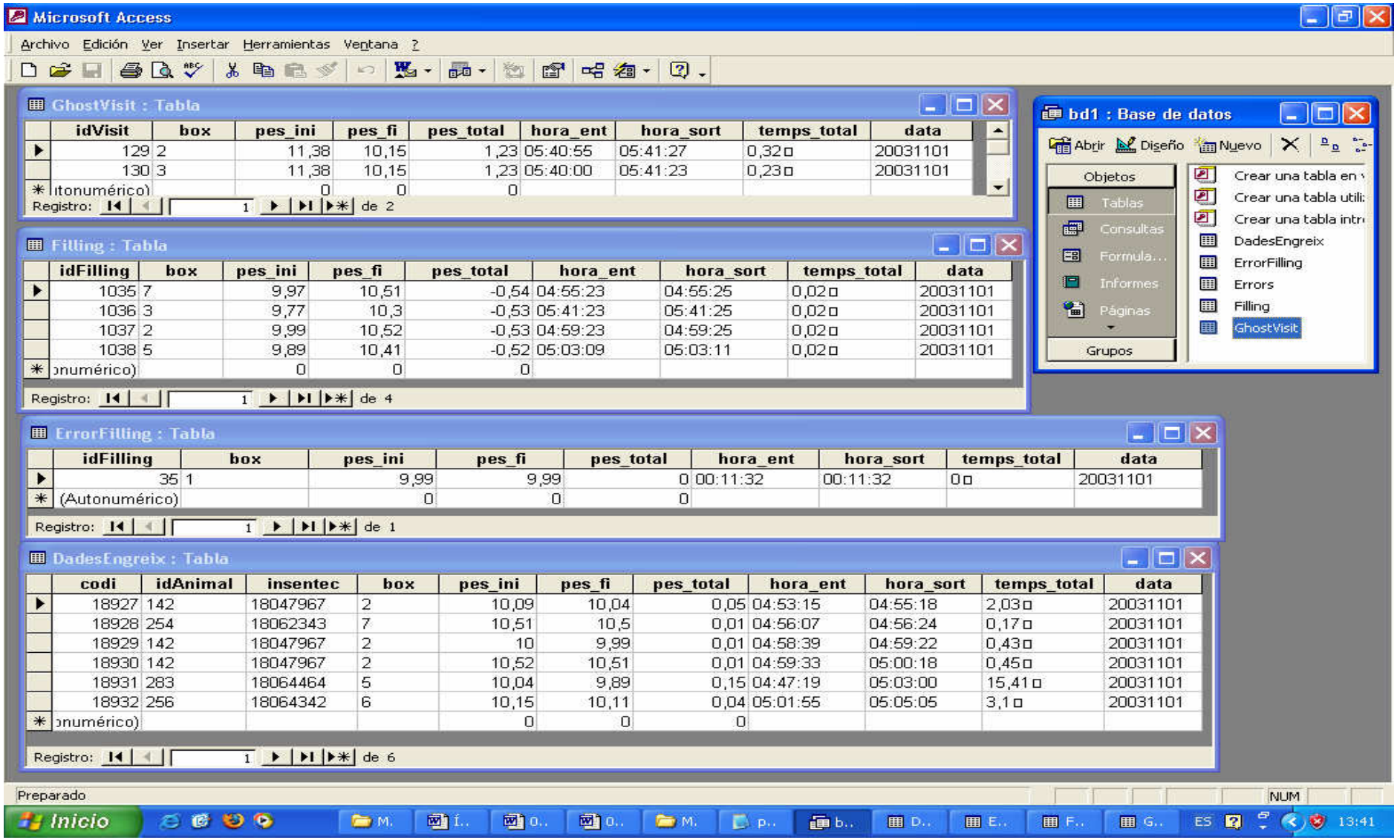


Figura 7.3: Taules de la BD amb les dades rebudes correctament del programa.

7.2. Eina de simulació amb *DataSocket*.

Les proves realitzades per comprovar l'eficiència del *DataSocket* en aquest treball són les següents:

- Provar la integritat i l'eficiència del codi mitjançant proves d'enviament i rebuda de dades reals obtingudes de la menjadora.

- Comprovar que la inserció a la BD remotament utilitzant el protocol *DataSocket* de *LabVIEW* funciona correctament emmagatzemant les dades rebudes del programa.

En aquestes proves hem utilitzat els mateixos fitxers *.txt* que en l'anterior apartat, ja que es tracta d'un programa amb la mateixa finalitat, però simulant la forma en que es produirà la inserció a la BD quan s'implanti al sistema global d'alimentació automatitzada.

El programa *send.vi* llegirà les dades del *fitxer.txt*, línia a línia. Al mateix temps que les llegirà, anirà enviant a través de *DataSocket* al programa d'adquisició de dades, *recieve.vi*, en el qual s'aniran rebent les línies de dades i emmagatzemant-les a la BD de dades que s'hagi triat anteriorment.

Per començar a executar els programes s'ha de fer amb el següent ordre:

1. Executar el programa *DataSocket Server* de *LabVIEW*.
2. Executar el programa *recieve.vi*, triant la BD que es vol, i el canal de *DataSocket* que s'utilitzarà per comunicar-se amb l'altre programa.
3. Executar el programa *send.vi*, assegurant-se que tingui assignat el mateix canal de *DataSocket* que el programa amb el que es vol comunicar.

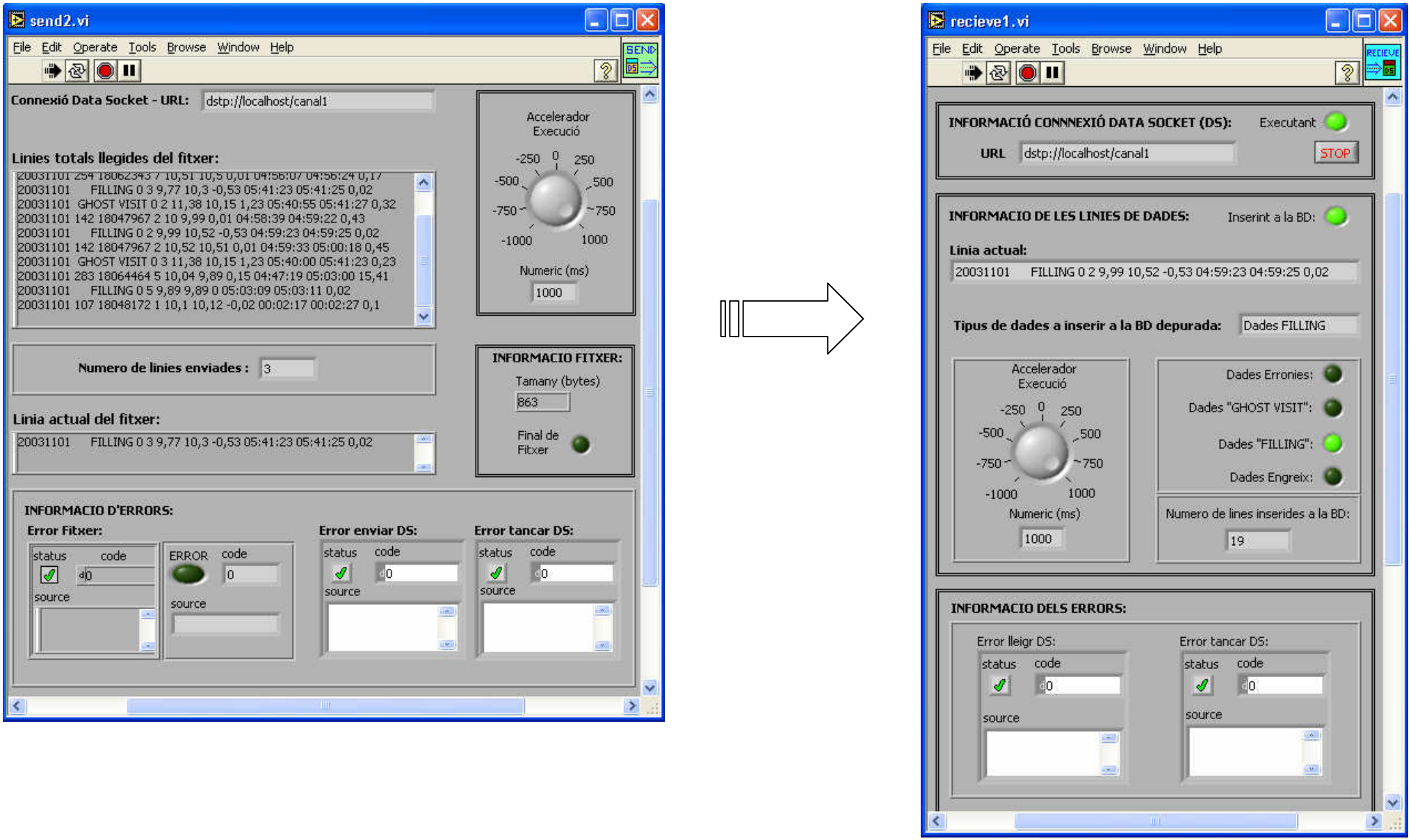


Figura 7.4: Simulació d'enviament i rebuda de dades amb els respectius *VIs*, *send* i *recieve*, executant-se simultàniament.

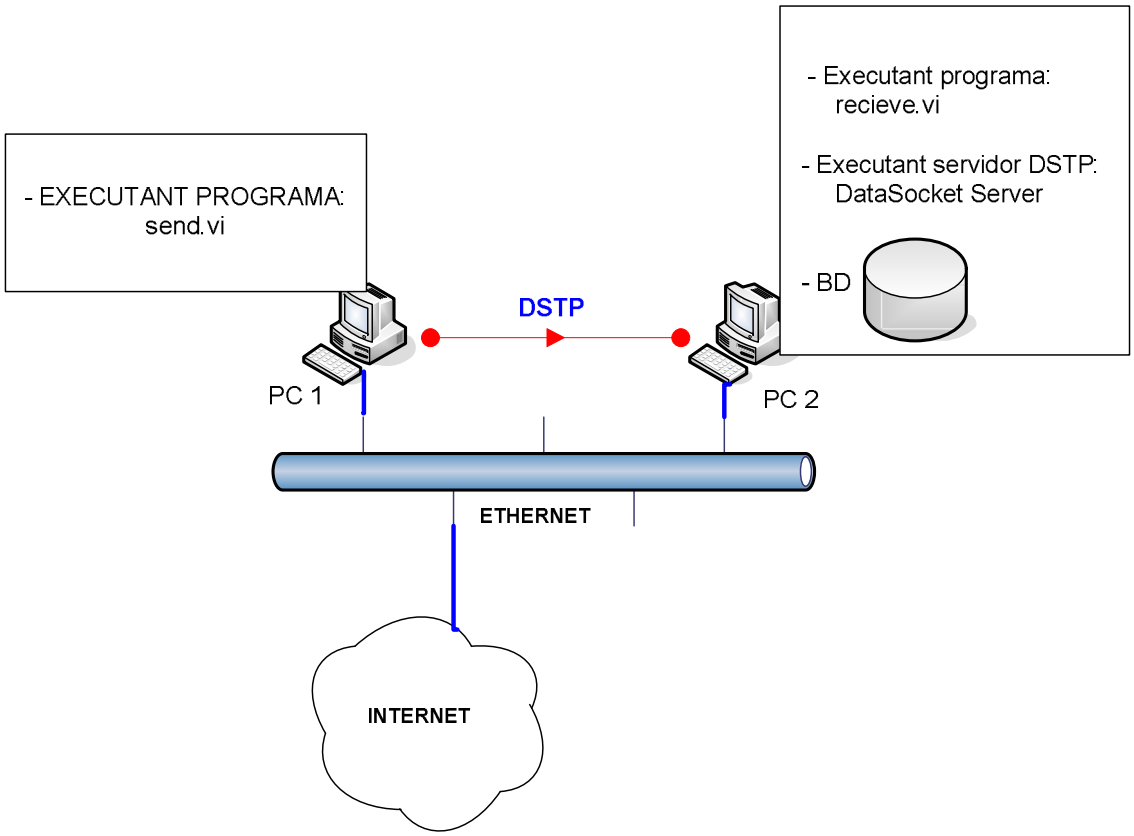


Figura 7.5-a: Diagrama del muntatge de simulació d'enviament i rebuda de dades.

A la *figura 7.5-a* es pot comprovar com està muntada la simulació amb *Data Socket Transfer Protocol* que a la vegada funciona a través de *ethernet*; com s'ha comentat al capítol 4, el protocol *DataSocket* està basat en *TCP/IP*.

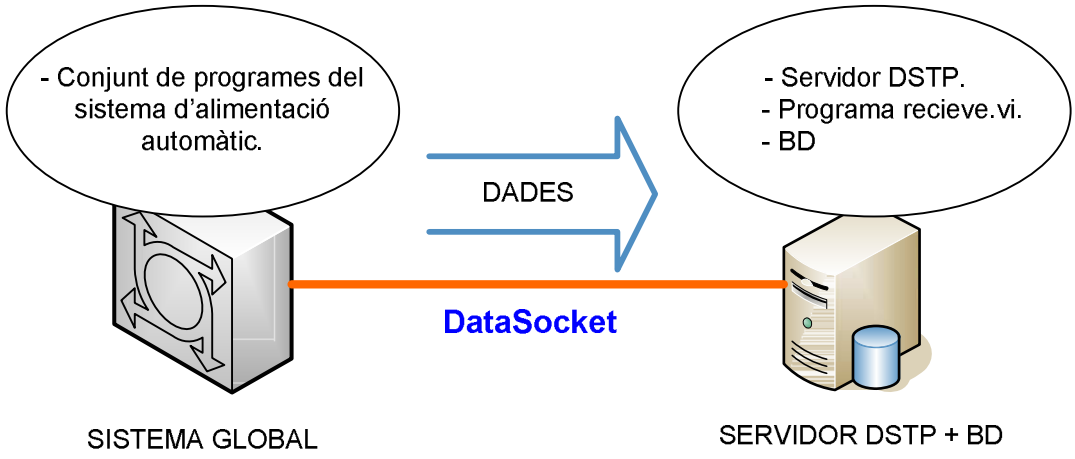


Figura 7.5-b: Diagrama d'integració del programa *recieve.vi* al sistema global.

A la *figura 7.5-b* es mostra com s'integrarà el programa *recieve.vi* amb el sistema global a través de *Data Socket Transfer Protocol*, que rebrà les dades i les anirà guardant automàticament a la respectiva BD. Hi haurà tants programes *recieve.vi* funcionant com BDs hi hagin al sistema, en aquest cas seran dos BDs. Això implica tenir tants canals *DataSocket* com BDs es vulguin.

Les proves realitzades s'han efectuat a partir dels fitxers de dades *.txt*, juntant 4 fitxers diaris per comprovar la robustesa i la fiabilitat dels programes. Cada fitxer tenia 3776 linies de dades, es a dir, 15104 linies de dades, enviades, rebudes, filtrades i emmagatzemades a la BD.

El resultat global de l'avaluació de la simulació amb el protocol *DataSocket* és satisfactori, tant el programa d'enviament (*send.vi*), com el de rebuda (*recieve.vi*), com la base de dades han funcionat de manera eficient.

7.3. Eina de consulta a la BD.

En aquest punt del treball s'explicaran les proves realitzades tant a les consultes efectuades de manera local com remota, per comprovar la eficiència del codi. Les consultes SQL seran les mateixes pels dos programes, amb l'afegit que al segon s'ha de tenir en compte la connexió en xarxa.

7.3.1. Consulta de dades localment.

Les proves efectuades per realitzar l'avaluació d'aquest programa són les següents:

- Diverses consultes a les BDs (correctes i incorrectes per provar el control d'errors).

- Reiniciar i/o parar el programa, amb la desconnexió i reconnexió a la BD que això suposa.

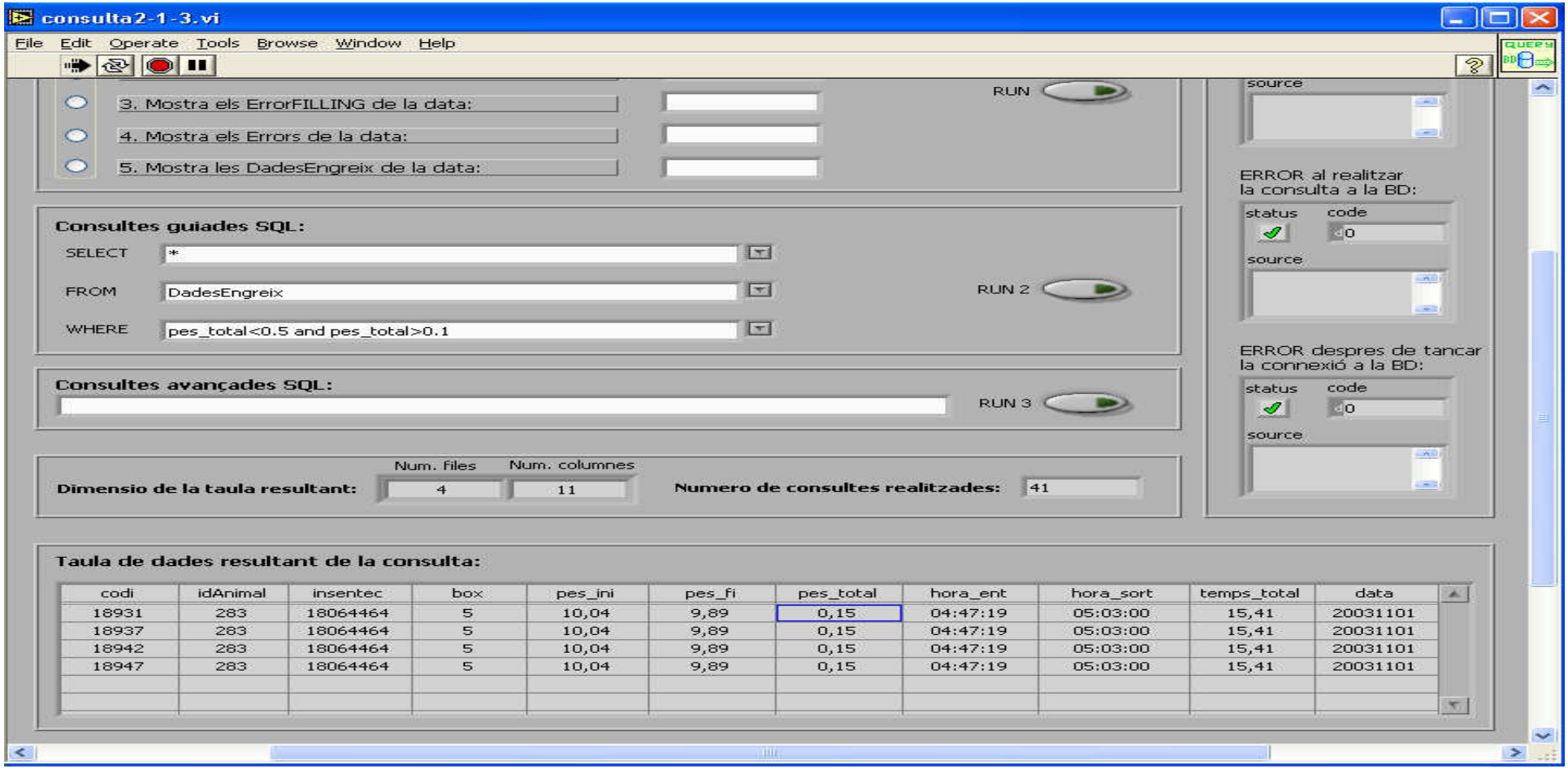
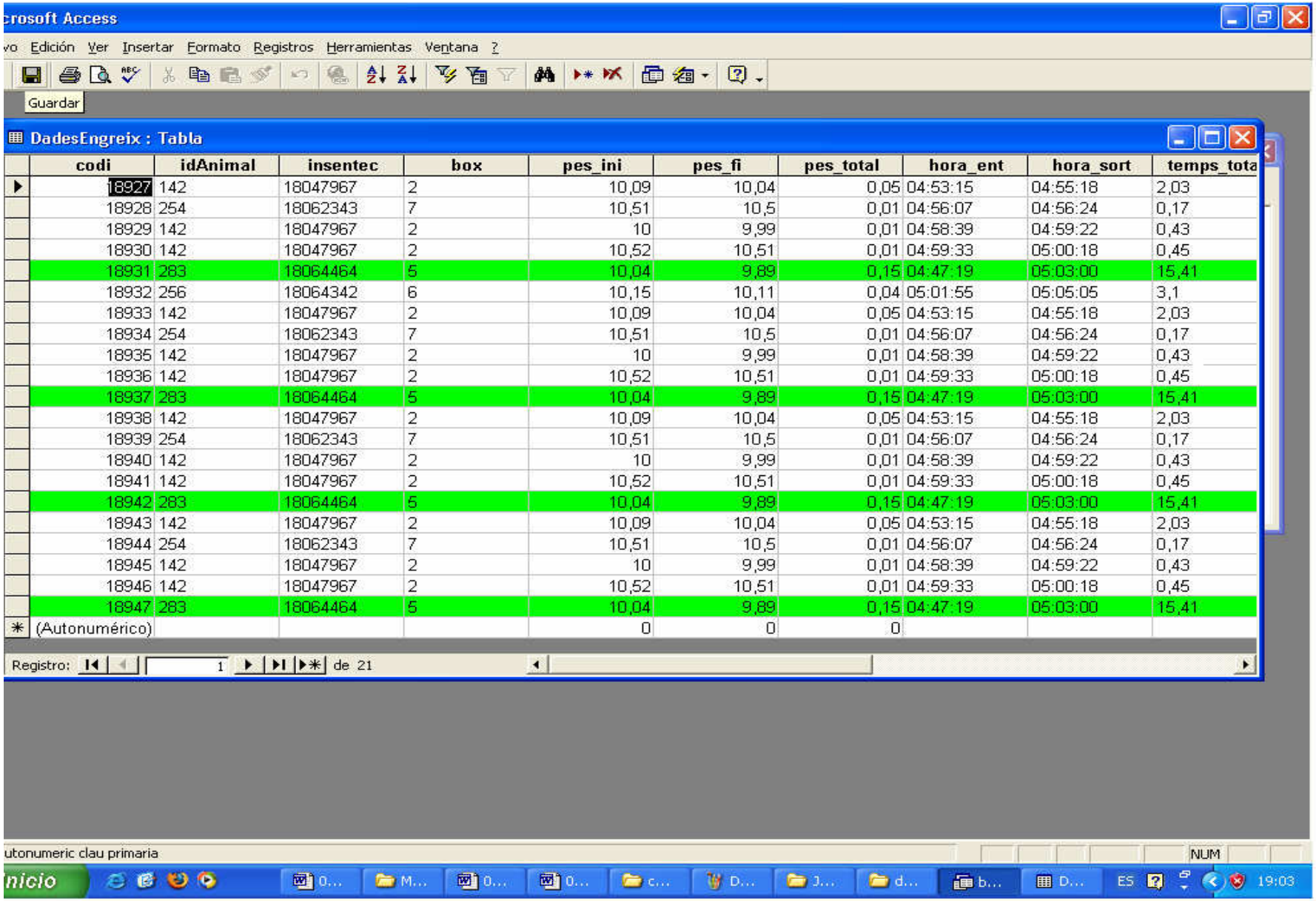


Figura 7.6-a: Resultats de la consulta a la BD per comparar amb les taules de la BD.

Com es pot observar comparant la *figura 7.6-a* i la *figura 7.6-b*, el resultat a la consulta: “SELECT * FROM DadesEngreix WHERE pes_total<0.5 ad pes_total>0.1” es correcte i obtenim un temps de resposta immediat.



codi	idAnimal	insentec	box	pes_ini	pes_fi	pes_total	hora_ent	hora_sort	temps_tota
18927	142	18047967	2	10,09	10,04	0,05	04:53:15	04:55:18	2,03
18928	254	18062343	7	10,51	10,5	0,01	04:56:07	04:56:24	0,17
18929	142	18047967	2	10	9,99	0,01	04:58:39	04:59:22	0,43
18930	142	18047967	2	10,52	10,51	0,01	04:59:33	05:00:18	0,45
18931	283	18064464	5	10,04	9,89	0,15	04:47:19	05:03:00	15,41
18932	256	18064342	6	10,15	10,11	0,04	05:01:55	05:05:05	3,1
18933	142	18047967	2	10,09	10,04	0,05	04:53:15	04:55:18	2,03
18934	254	18062343	7	10,51	10,5	0,01	04:56:07	04:56:24	0,17
18935	142	18047967	2	10	9,99	0,01	04:58:39	04:59:22	0,43
18936	142	18047967	2	10,52	10,51	0,01	04:59:33	05:00:18	0,45
18937	283	18064464	5	10,04	9,89	0,15	04:47:19	05:03:00	15,41
18938	142	18047967	2	10,09	10,04	0,05	04:53:15	04:55:18	2,03
18939	254	18062343	7	10,51	10,5	0,01	04:56:07	04:56:24	0,17
18940	142	18047967	2	10	9,99	0,01	04:58:39	04:59:22	0,43
18941	142	18047967	2	10,52	10,51	0,01	04:59:33	05:00:18	0,45
18942	283	18064464	5	10,04	9,89	0,15	04:47:19	05:03:00	15,41
18943	142	18047967	2	10,09	10,04	0,05	04:53:15	04:55:18	2,03
18944	254	18062343	7	10,51	10,5	0,01	04:56:07	04:56:24	0,17
18945	142	18047967	2	10	9,99	0,01	04:58:39	04:59:22	0,43
18946	142	18047967	2	10,52	10,51	0,01	04:59:33	05:00:18	0,45
18947	283	18064464	5	10,04	9,89	0,15	04:47:19	05:03:00	15,41
(Autonumérico)				0	0	0			

Figura 7.6-b: Taula de la *BD-neta* on es mostra marcats en verd el resultat de la consulta realitzada a les proves.

7.3.2. Consulta de dades remotament.

Les proves efectuades per realitzar l'avaluació d'aquest programa són les següents:

- Comprovar que la connexió al servidor és bona i ens garanteix un bon funcionament del programa podent realitzar la petició de control al *VI* des d'una màquina remota.

- Diverses consultes a les BDs (correctes i incorrectes per provar el control d'errors).

- Reiniciar i/o parar el programa remotament, mantenint la connexió amb el servidor.

Un cop configurat el *webServer* de *LabVIEW* i s'ha publicat la *web* del *VI* que es vol manipular, ja es pot demanar el control remot (*request control*) del *VI* en qüestió, sempre i quan s'hagi alliberat del servidor que es qui té el control per defecte (*unlock control*).

Com es pot observar a la *figura 7.7* es pot manipular completament el *VI* de consulta a la BD de manera remota a través de la xarxa, tant parar, com reiniciar el programa, com fer les consultes que es vulguin, com desconnectar i connectar a una altra BD... Es treballa pràcticament igual que en local, l'únic que es fa a mitjançant un *navegador web* i connexió a xarxa.

El resultat global de l'avaluació és satisfactori, tant el programa de consultes de manera local a la base de dades, com de manera remota han funcionat correcta i eficientment. També s'ha comprobat el funcionament del *WebSever* de *LabVIEW* que ha sigut correcte, mitjançant el protocol *TCP/IP* a través d'una xarxa, utilitzant programes navegadors a les màquines remotes (clients).

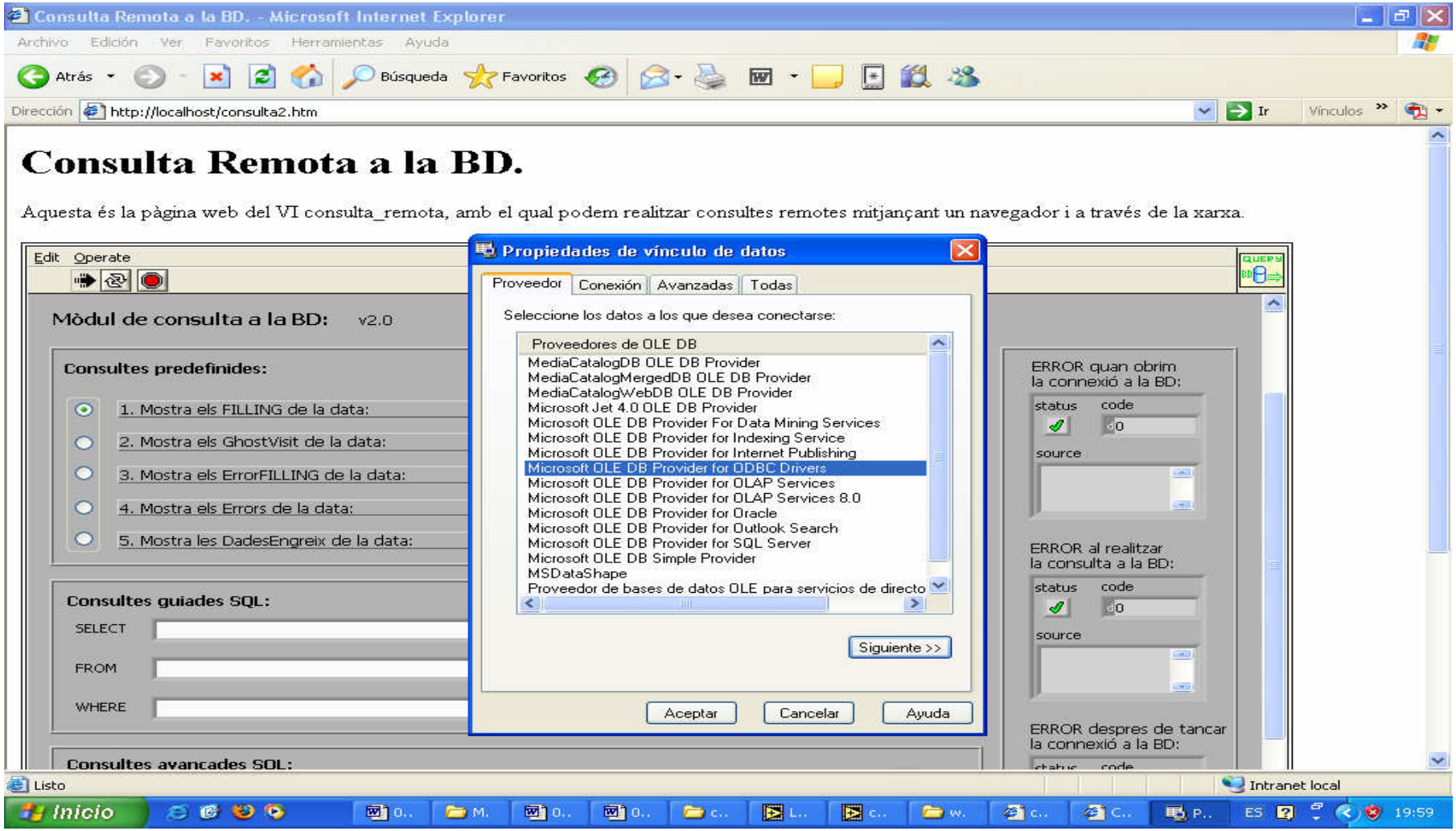


Figura 7.7: Demostració de executar i parar el programa remotament mantenint el control de *VI* i la connexió amb el servidor.

7.4. Eina de generació d'informes amb *Report Generation*.

A continuació es mostraran les proves realitzades dels últims programes d'aquest treball. Els programes en qüestió són els de generació d'informes, tant de manera local com remota, i es pretén comprovar la eficiència del codi.

Consum total (Kg) 0,28

codi	idAnimal	insentec	box	pes_ini	pes_fi	pes_total	hora_ent	hora_sort	temps_total	data
18927	142	18047967	2	10,09	10,04	0,05	4:53:15	4:55:18	2,03	20031101
18929	142	18047967	2	10	9,99	0,01	4:58:39	4:59:22	0,43	20031101
18930	142	18047967	2	10,52	10,51	0,01	4:59:33	5:00:18	0,45	20031101
18933	142	18047967	2	10,09	10,04	0,05	4:53:15	4:55:18	2,03	20031101
18935	142	18047967	2	10	9,99	0,01	4:58:39	4:59:22	0,43	20031101
18936	142	18047967	2	10,52	10,51	0,01	4:59:33	5:00:18	0,45	20031101
18938	142	18047967	2	10,09	10,04	0,05	4:53:15	4:55:18	2,03	20031101
18940	142	18047967	2	10	9,99	0,01	4:58:39	4:59:22	0,43	20031101
18941	142	18047967	2	10,52	10,51	0,01	4:59:33	5:00:18	0,45	20031101
18943	142	18047967	2	10,09	10,04	0,05	4:53:15	4:55:18	2,03	20031101
18945	142	18047967	2	10	9,99	0,01	4:58:39	4:59:22	0,43	20031101
18946	142	18047967	2	10,52	10,51	0,01	4:59:33	5:00:18	0,45	20031101

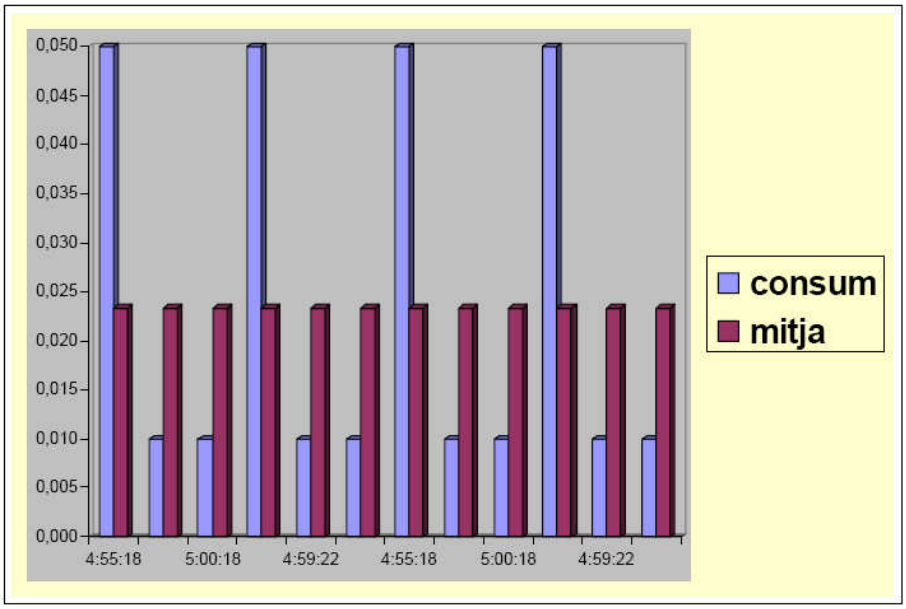


Figura 7.8: Exemple d'informe generat pel programa de generació d'informes.

7.4.1. Generació d'informes localment.

A la *figura 7.8* es pot observar un informe realitzat amb el programa de generació d'informes de manera local, amb les característiques següents:

- Animals del *box* 2.
- Per tota la experiencia (tots els dies registrats).
- Consum total (característica principal de l'informe).
- Gràfica de barres 3D.

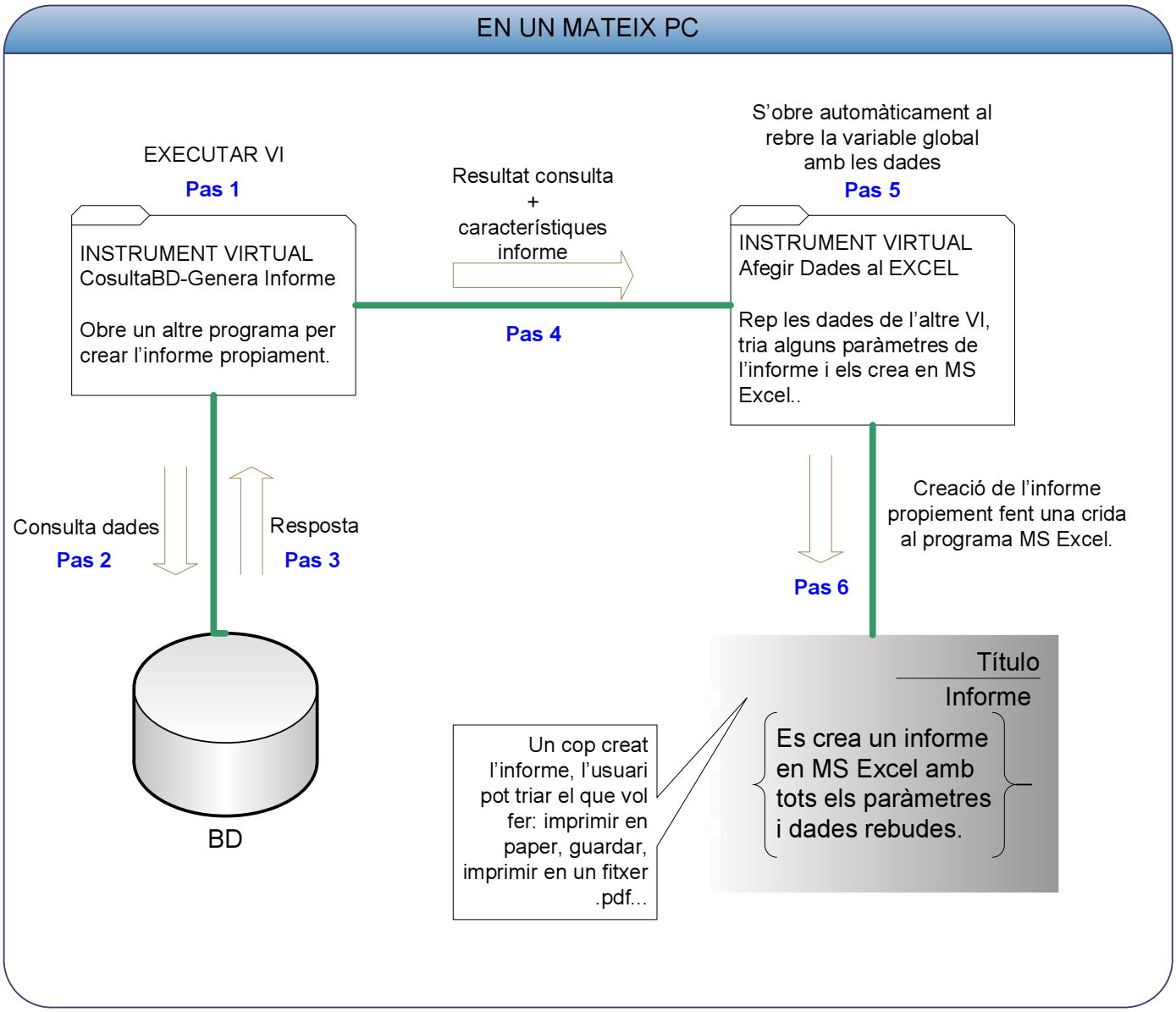


Figura 7.9: Diagrama de generació de l'informe.

El resultat de l'informe (*figura 7.8*) indica en primer lloc un requadre on es troba la característica principal de l'informe, a mode de títol, en aquest exemple és "Consum total" expressat en Kg, equival a la suma de tot el menjar consumit pels animals escollits, en aquest cas 0.28 Kg. A continuació es pot observar la taula resultant i per últim la gràfica, en aquest exemple mostra el consum de cada animal individualment i la mitja d'aquests consums.

A la *figura 7.9* s'explica l'esquema detallat del procés que s'ha de seguir per generar un informe:

- **Pas 1**: S'executa el programa *consultaBD-generaInforme.vi,* realitzant la connexió a una BD corresponent. En aquest *VI* existeixen dues pestanyes, una de consulta normal i l'altre de consulta associada a informe predissenyats, s'escolleix la segona.

- **Pas 2**: Realitzar una consulta.

- **Pas 3**: Es mostra la consulta en una taula de dades, i es fa click al botó de generació d'informes, al final del pannell frontal.

- **Pas 4**: Passa les dades al programa *afegirDadesAlExcel.vi* mitjançant una variable global.

- **Pas 5**: S'obre automàticament el programa *afegirDadesAlExcel.vi* amb referències des de l'altre programa. Es trien uns paràmetres del futur informe (característiques gràfica, nom informe...) i es fa click al botó generar informe, el qual passarà totes les dades al *MS Excel*.

- **Pas 6**: S'executa el programa *MS Excel* de forma automàtica i es crea l'informe amb tots el paràmetres rebuts.

Les proves efectuades per realitzar l'avaluació d'aquest programa són les següents:

- Generació de diversos informes a partir de consultes a la BD.

- Comprovar que es comuniquen correctament els programes de consulta amb el de generació d'informes.
- Comprovar que l'informe final és correcte, tant la gràfica com la taula de dades.

7.4.2. Generació d'informes remotament.

Les proves efectuades per realitzar l'avaluació d'aquest programa són les següents:

- Comprovació de la connexió en xarxa entre el client i el servidor (petició de control).
- Generació de diversos informes a partir de consultes a la BD remota, i guardar-lo a la màquina client.
- Comprovar que l'informe final és correcte, tant la gràfica com la taula de dades, i que s'ha desat al PC client correctament.

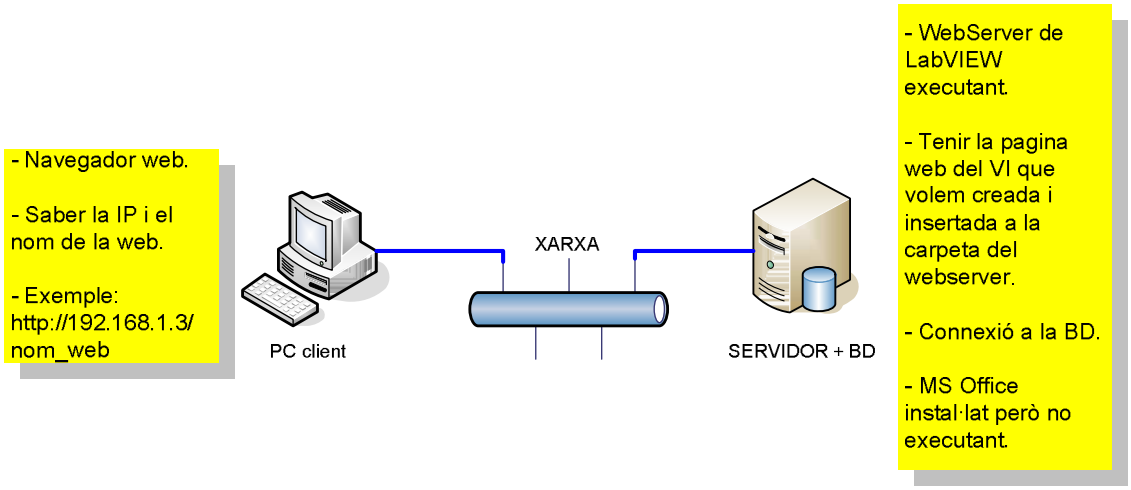


Figura 7.10: Diagrama de funcionament i de la connexió de l'informe remot.

El resultat global de l'avaluació és satisfactori, tant el programa de generació d'informes de manera local, com de manera remota han funcionat correcta i eficientment. També s'ha comprobat el funcionament del *WebSever* de *LabVIEW* que ha sigut correcte, mitjançant el protocol *TCP/IP* a través d'una xarxa, utilitzant programes navegadors a les màquines remotes (clients). També s'ha tingut en compte a l'avaluació el programa per mostrar l'informe final, que en aquest cas ha sigut *MS Excel 2003* i ha funcionat correctament.

8 Conclusions

Capítol 8

Conclusions.

Les conclusions extretes d'aquest treball són les següents:

- L'entorn de programació ***LabVIEW*** ha permès la implementació de varies eines per a l'anàlisi, filtrat, emmagatzemament, inspecció i consulta de dades i generació d'informes, d'un gran volum de dades procedents d'un sistema d'alimentació automatitzada per una explotació porcina.

- La implementació de programes individuals i independents entre si (instruments virtuals de *LabVIEW*), ha permés acoblar les eines realitzades en aquest treball a un sistema s'alimentació automàtic implemetat en *LabVIEW*, i anar-lo ampliant, si es requereix, sense fer canvis en els programes actuals.

- La realització d'un filtrat previ i la posterior estructuració en les BDs, de les dades procedents d'un sistema d'alimentació automatitzada, ha resultat molt útil per a la posterior inspecció de les dades tant en consultes a la BD com en informes.

- La tècnica del prototipatge utilitzada per al disseny i la implementació en llenguate *G* de *LabIVEW* dels diferents programes, ha ajudat a treballar de manera iterativa i progressiva durant l'elaboració de cada eina. Així doncs, en les evolucions de les diferents eines, es pot reaprofitar bona part del codi i ampliar-la amb relativa facilitat.

- La **BD** creada en *MS Access 2003*, ha resultat eficient en tot moment, tant en el procés d'emmagatzematge, com de consulta mantenint les dades persistentment i interaccionant amb el programa implementat en *LabVIEW* de manera correcta.

- Els ***toolkits*** de *LabVIEW* emprats en aquest treball, *DataBase Toolkit* i *Report Generation Toolkit (for MS Office),* han resultat de gran ajuda per la implementació de les parts dels programes que interaccionaven tant amb bases de dades com amb generació d'informes; encara que, el *Report Generation* és una mica rígid alhora de fer certes parts dels informe tal com operacions matemàtiques, taules de dades... que s'han hagut de fer previament amb *LabVIEW*, per després passar el resultat directament a l'informe final.

- El ***webServer*** de *LabVIEW* ha funcionat correctament, i ha facilitat l'accés remot als diferents programes que ho requerien en aquest treball; encara que, només permet el control simultani a un PC, ja sigui remot o el mateix servidor, demanant el control previament.

- El ***DataSocket Transfer Protocol*** de *LabVIEW* ha permès comunicar varis programes remotament, simulant així la comunicació real que existirà entre els diferents instruments virtuals (programes) al sistema global.

8.1. Línies futures de treball.

En un treball futur es podria realitzar altres accions com, adaptar el sistema amb un gestor de base de dades més potent, com per exemple *PostgreSQL, MySQL, SQLserver*... per assegurar que no hi existirà falta d'espai alhora d'omplir les taules.

També es podria implementar un altre programa generador d'informes utilitzant una altra tecnologia que no fos *LabVIEW*, però que si fos compatible amb el sistema global d'alimentació automatitzada. Algun programa que ho permetés podria ser *DIAdem* o *Crystal Reports,* aquest últim més utilitzat al mercat, però l'avantatge del primer és que pertany a la mateixa companyia que *LabVIEW, National Instruments*, això ens ofereix una certa confiança i ens assegura en bona part la compatibilitat amb el sistema global implementat en *LabVIEW* i utilitzant eines i *toolkits* pròpies de la plataforma.

9 Bibliografia

Bibliografia.

BALTER, A. (2004). "Manual Fundamental de Microsoft Office Access 2003". 1ª Edició. Madrid. ISBN 84-415-1658-8.

BISHOP, Robert H. (1999). "*Learning with LabView*". Menlo Park, California: Addison Wesley, cop. 1999. ISBN 0-201-36166-3.

COULOURIS, G; DOLLIMORE, J; KINDBERG, T. (2001). "Sistemas distribuidos. Conceptos y diseño". Madrid, Addison Wesley. ISBN 84-7829-049-4.

FARACO, G; Gabrielle L., "*Using LabVIEW for applying mathematical models in representing phenomena*", Computer & Education, 96-112, 2006.

INSENTEC, [online] *website* oficial de la empresa que subministra menjadores d'animals: http://www.insentec.nl

MANUEL LAZARO, Antonio. (2001). "LabView 6i: programación gráfica para el control de instrumentación". Madrid, Paraninfo. ISBN 84-415-1658-8.

MANUEL LAZARO, Antonio; del RIO FERNÁNDEZ, Joaquín. (2005). "LabView 7.1: programación gráfica para el control de instrumentación". Madrid, International Thomson Paraninfo, cop. ISBN 84-9732-391-2.

MICROSOFT, [online] Informació sobre *ODBC*. http://msdn2.microsoft.com/en-us/library/ms710252.aspx

NATIONAL INSTRUMENTS, [*CD-ROM*] Material audiovisual interactiu: "*Basics / Interactive Training (Disk I & Disk II)*.

NATIONAL INSTRUMENTS, [Manual oficial] "*Database Connectivity Toolset User Manual ", May 2001 Edition Part Number 321525C-01*.

NATIONAL INSTRUMENTS, [Manual oficial] *"LabVIEW Report Generation Toolkit for Microsoft Office User Guide"*.

NATIONAL INSTRUMENTS, [Tutorial oficial] *"Creating a Report in Microsoft Excel Using the LabVIEW Report Generation Toolkit"*.

NATIONAL INSTRUMENTS, [online] *website* oficial de la empresa nord americana *National Instruments*: http://www.ni.com/.

NATIONAL INSTRUMENTS, [online] NI Discussion Forums: http://www.forums.ni.com

NATIONAL INSTRUMENTS, [online] NI Developer Zone: http://zone.ni.com/devzone/cda/main

PRESSMAN, Roger S. (1993). "Ingenieria del Software. Un enfoque práctico". 3a edició. Madrid [etc.] : McGraw-Hill, cop. ISBN 84-481-0026-3.

RIERA, N. (2006). "Anàlisi, disseny i avaluació d'una eina informàtica d'ajuda a l'anàlisi i inspecció de dades d'alimentació de porcs d'engreix, capaç de generar informes automàticament". PFC - UdL.

SOMMERVILLE, Ian. (1992). "*Software engineering*". 4a. edició. Londres [etc.] : Addison-Wesley, cop. ISBN 0-201-56529-3.

WIKIPEDIA, [enciclopèdia online] http://www.wikipedia.org/

10 Annexes

Annex A.

DSN.

El *Data Source Name* (DSN) o Nom d'Origen de Dades, són uns noms o referències a les bases de dades que utilitzen els sistemes *Windows* per traballar amb aquestes *BDs* per connexió *ODBC*.

Els *DSN* representen tot el relacionat a una font de dades configurada per l'usuari per connectar-se a les *BDs,* i s'especifiquen les dades que necessita *Windows* per treballar amb dites *BDs* (nom del servidor o origen de dades, cadena de connexió...) . Es a dir, per cada connexió que l'usuari requereixi establir amb algun fabricant, té que especificar una sèrie d'informació que permetin al controlador o *driver* saber amb quin fabricant s'ha de connectar i la cadena de connexió que l'ha d'enviar per establir la connexió amb la font de dades *ODBC* accedida pel proveïdor en qüestió.

Dins de l'*ODBC,* s'haurà de crear un DSN de tipus sistema o usuari. Això es realitza desde el *Panel de Control de Windows, Herramientas Administrativas, Origenes de datos ODBC.* S'hauràn d'afegir tants *DSNs* com connexions a bases de dades diferents es vulgui tenir, i per cada un d'ells es seleccionarà el driver de la aplicació que hem utilitzat per crear la *BD,* el nom que se li vulgui donar i el *path* o la ruta on es troba al disc dur.

Un origen de dades d'usuari *ODBC* emmagatzema informació de connexió al proveïdor de dades indicat. Un origen de dades d'usuari només és visible i utilitzable en l'equip actual i per l'usuari indicat.

Un origen de dades de sistema *ODBC* emmagatzema informació de com connectar-se al proveïdor de dades indicat. Un origen de dades de sistema és visible per tots els usuaris d'aquest equip, inclosos els serveis *NT*.

Aquest *DSN,* el que permet és definir la base de dades que serà interrogada, sense necessitat de passar per l'aplicació que s'hagi utilitzat per crear-la, es a dir, amb simples

crides i ordres des d'un programa es pot obtenir les dades que es busquen sense necessitat d'executar el *SGBD* (per exemple: *MS Access 2003*), els quals no tenen perque trobar-se en el servidor on s'està treballant.

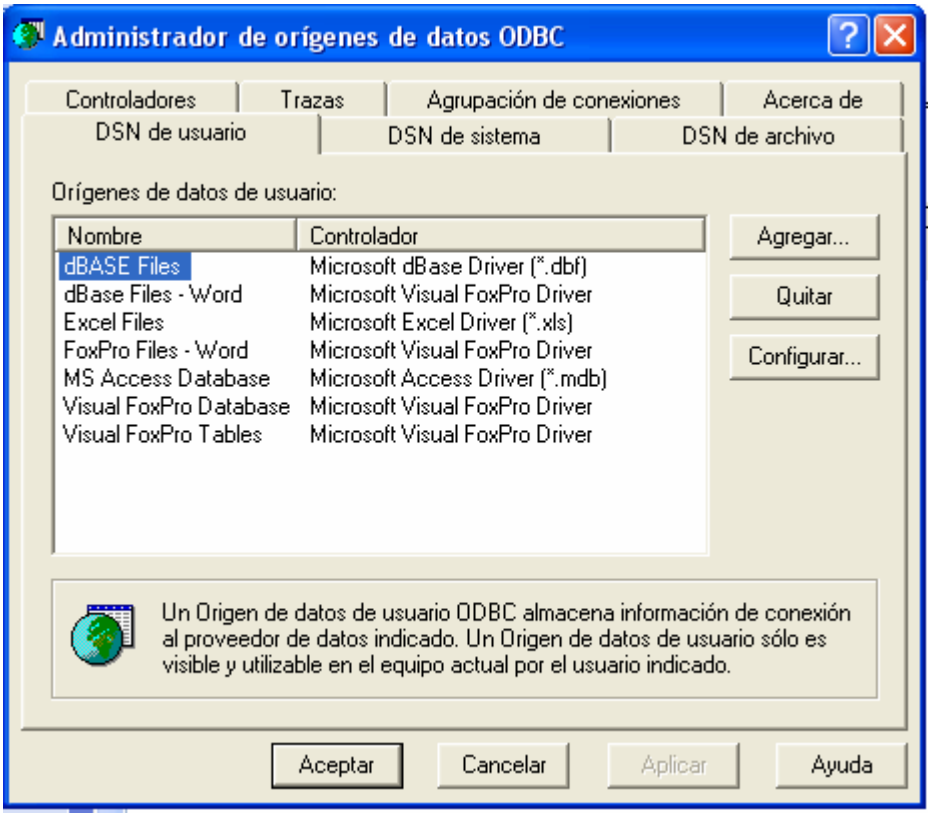


Figura 10.A: Administrador de dades de *ODBC*.

Annex B.

ODBC.

Open DataBase Connectivity (*ODBC*) és un estàndard d'accés a bases de dades desenvolupat per l'empresa *Microsoft Corporation*, l'objectiu del qual es fer possible accedir a qualsevol dada de qualsevol aplicació, sense importar quin Sistema Gestor de Base de Dades (*SGBD*) emmagatzemi les dades.

ODBC aconsegueix això al insertar una capa intermitja anomenada "manegador" de bases de dades entre l'aplicació i el *SGBD*. El propòsit d'aquesta capa és traduir les consultes de dades de l'aplicació en comandes que el *SGBD* entengui.
Per a que això funcioni tant l'aplicació com el *SGBD*, han de ser compatibles amb *ODBC*, això vol dir que l'aplicació ha de ser capaç de produir comandes ODBC i el SGBD ha de ser capaç de respondre a elles.

Per connectar-se a la base de dades es crea un *DSN* dins de l'*ODBC* que defineix els paràmetres, ruta i característiques de la connexió, segons les dades que sol·liciti el fabricant.

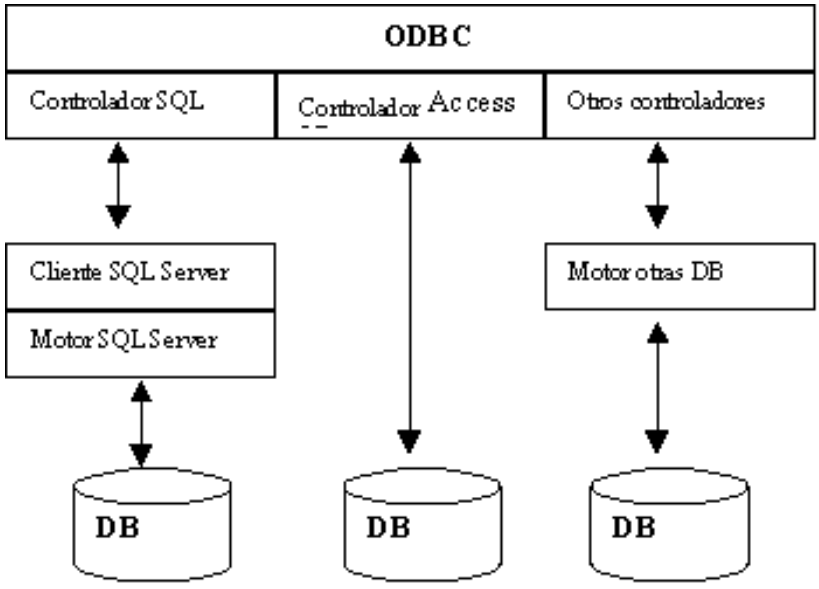


Figura 10.B: Esquema de diferents controladors *ODBC*.

En l'actualitat existeixen *ODBC* per molts sistemes gestors de bases de dades, tal com *Informix, MS Access, PostgreSQL, MySQL, SQL Server*... i els fabricants de les diferents bases de dades són els responsables de crear un *driver ODBC* per a que la seva *BD* es pugui connectar des d'un sistema *Microsoft.*

Annex C.

PRESSUPOST.

Pressupost

PROJECTE	Desenvolupament d'un sistema d'adquisició i gestió de dades integrat en una arquitectura modular en xarxa (orientat a la ramaderia de precisió)

	Preu / hora	Hores	Subtotal
Analista			
Anàlisi i requeriments	39,00	30	1.170,00
Dissenys preliminars	39,00	30	1.170,00
Anàlisi conceptual BD	39,00	30	1.170,00
Disseny de la BD bruta	39,00	20	780,00
Disseny de la BD neta	39,00	20	780,00
Disseny de la comunicació	39,00	30	1.170,00
Total analista	**39,00 €**	**160**	**6.240,00 €**

	Preu / hora	Hores	Subtotal
Programador			
Disseny interficie	29,00	60	1.740,00
Implementació	29,00	340	9.860,00
Total programador	**29,00 €**	**400**	**11.600,00 €**

TOTAL		**17.840,00 €**

El pressupost realitzat en aquest treball és un càlcul estimatiu de les hores emprades per realitzar el mateix. S'han contemplat les hores de disseny i de implementació de l'aplicació pròpiament, prenent com a referència els dies dedicats a cada part.

Per l'anàlisi i requeriments en general, es a dir, la tasca com analista, s'han comptabilitzat 160 hores, calculant que s'ha estat 2 mesos a mitja jornada (4 hores diàries), són 80 hores al mes.

Per la part de la implementació, es a dir, la tasca com programador, s'han comptabilitzat 400 hores, calculant que s'ha estat 5 mesos a mitja jornada (4 hores diàries), són 80 hores al mes.

En aquest pressupost no s'han comptabilitzat les hores d'aprenentatge del llenguatge de programació, ni de les diferents proves que s'han anat realitzant durant tot el transcurs del treball, ni de la redacció de la memòria, ni aspectes d'organització del treball en general.